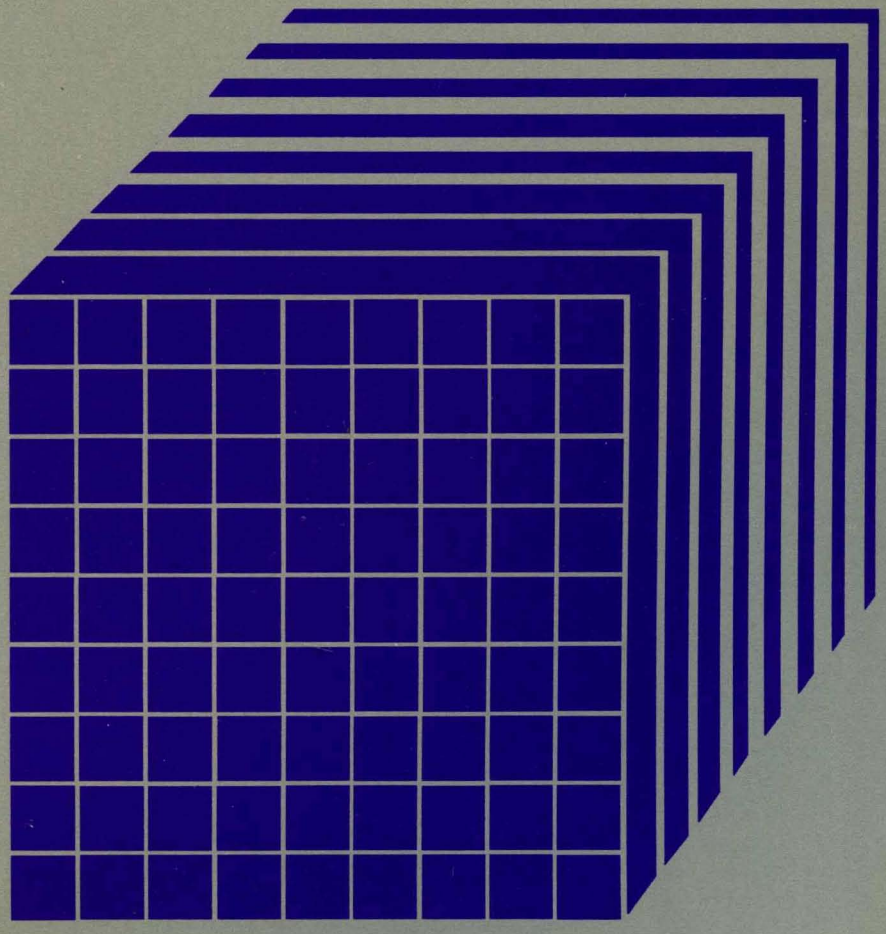
IBM

Virtual Machine/ System Product

Service Routines Program Logic

Release 5

LY20-0890-3



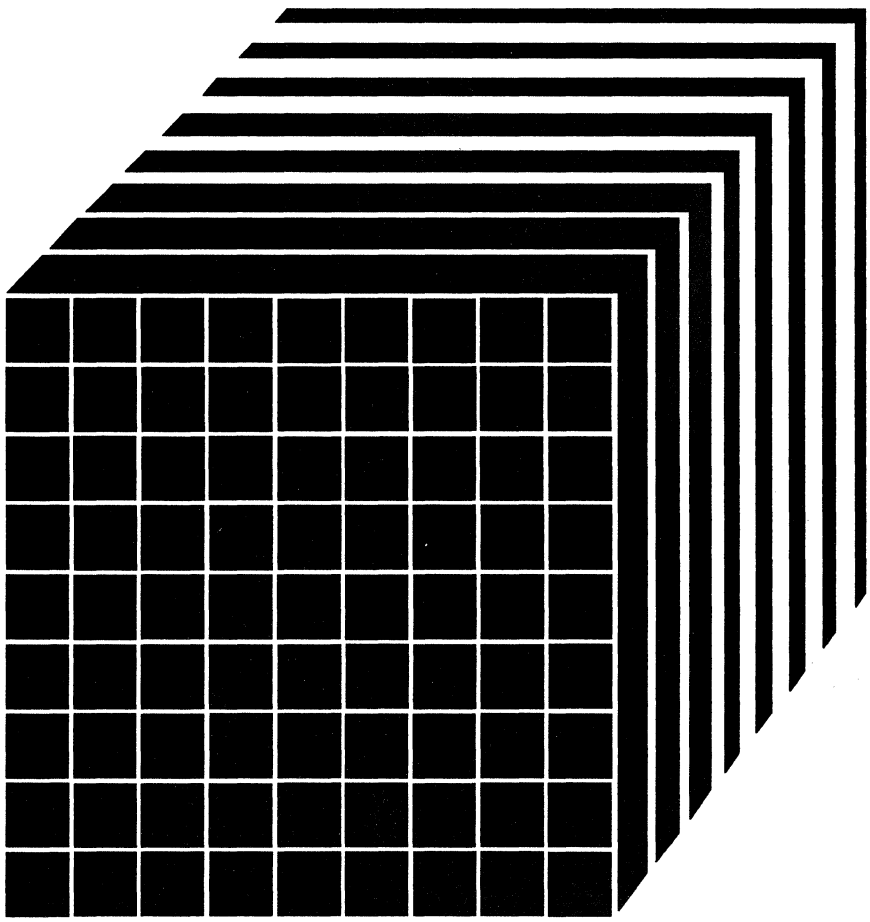
IBM

Virtual Machine/ System Product

Service Routines Program Logic

Release 5

LY20-0890-3



Fourth Edition (December 1986)

This edition, LY20-0890-3, is a major revision of LY20-0890-2. It applies to Release 5 of Virtual Machine/System Product (VM/SP), program number 5664-167, and to all subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information contained herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Summary of Changes

For a list of changes, see page 275.

Technical changes or additions to the text and illustrations are indicated by a vertical bar to the left of the change.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

Ordering Publications

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department G60, P.O. Box 6, Endicott, NY, U.S.A. 13760. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This publication is intended for programmers responsible for updating VM/SP service routines. It explains the program logic for each of the VM/SP service routines. Because the service routines are unrelated, they are discussed separately. One chapter of this publication is dedicated to each service routine (or logical group of service routines).

A brief index is included at the end of each chapter.

The **Format/Allocate Service Program** chapter describes the program that formats disks so that they can be used by VM/SP.

The **Directory Program** chapter describes the program that creates the VM/SP directory.

The **DASD Dump Restore Program** chapter describes the program that dumps, restores, and copies system disk files.

The **Installation Verification Procedure** chapter describes the EXEC procedure that checks the accuracy of the starter or newly generated system.

The **Procedures for Generating and Updating VM/SP** chapter describes the EXEC procedures and modules that apply updates to the system, load the system, and generate new macro libraries.

The **VM/SP Starter System** chapter describes the system that is distributed to be used for system generation.

The **3704/3705 Service Programs** chapter describes the programs that perform generation and service functions for the control program for the IBM 3704/3705 Communications Controllers.

The **ZAP Service Program** chapter describes the program that modifies and dumps MODULE, LOADLIB, and TXTLIB files.

The **EREP/Error Recording Interface** chapter describes the modules that interface between CMS and the OS/VS EREP program.

The **MSS Communicator** chapter describes the program that operates in a virtual machine under OS/VS and interfaces between VM/SP and the MSS Mass Storage Control.

The **IEBIMAGE Interface** chapter describes the utility programs required to dynamically change the character arrangement tables, graphic modifications, copy modifications, and FCBs for the 3800 Printing Subsystem.

The **Command Class Override Program** chapter describes the program that builds an internal class-override file.

Contents

Figures

Chapter 1. Format/Allocate

Introduction

The Format/Allocate service routine is a standalone program for IBM 2305 Series, 2314, 2319, 3330, 3340, 3350 Series, 3375, 3380, and FB-512 Direct Access Storage Devices. The Format/Allocate program can be used to:

- Format all or part of a DASD device
- Allocate DASD space
- Create volume labels.

Operands entered from the IPL device and/or a console control the execution of the Format program.

With the inclusion of FB-512 devices, the format/allocate service program now supports two distinct types of DASD devices. It is important to understand the differences in these types.

The main difference is one of data format and addressing. One type, count-key-data, is referenced by a cylinder, head, and record number. A given record has two components; a count field and a data field. The count field contains the DASD address (cchhr) and length of the corresponding data. Formatting for CP's use means that these count and data fields are initialized to 4096-byte records (format writes 4096-byte records).

The other type, FB-512 devices, are addressed by a block number. The data is thought of as a linear address space of n blocks, numbered 0 through n-1. Each block is 512 bytes of data. Therefore, a CP page consists of eight consecutive blocks. Because the data is not stamped with a self-identifying label (such as cchhr in the count field of count-key-data devices), and the length of each block is fixed, the concept of formatting is quite different. Count-key-data space is formatted and allocated in units of cylinders. That means that the user "talks" to format/allocate by referring to specific cylinder numbers. FB-512 disk space is formatted and allocated in units of pages.

The distinction between count-key-data and FB-512 operation is detailed in the following pages.

If you install the speed matching buffer feature (Feature #6550) with the 3380, the extended count-key-data channel programs are used.

Format Operation

Count-Key-Data

The Format program writes 4096-byte (one page) records on all the specified cylinders. The records just written are then read to verify the disk surface. Any records not passing the read-after-write check are counted. When the format operation is complete, a summary of the addresses of the unusable pages is written on the console.

The first three records of cylinder 0 contain special system data including the volume label. If the format operation includes cylinder 0, any existing volume label is read first and if an OS Format 4 label is present, the information in the label concerning alternate track assignments is carried forward to the new label. Then the new volume label is written on the DASD device.

If cylinder 0 is not to be formatted, label checking is performed.

If unrecoverable DASD errors occur during the formatting operation, the format function is canceled, the message

DMKFMT735E FATAL DASD I/O ERROR CSW = csw

is issued, and the next control statement is read.

FB-512

The Format program writes zeros in the specified pages. The write is done with a read-back check to verify the disk surface. The format operation stops if any block fails the read-after-write check. The error message contains the block number in error.

The first two pages (pages 0 and 1) contain special system data including the volume label. If the format operation includes page 0, a volume label is written. If page 0 is not to be formatted, label checking is performed.

If unrecoverable DASD errors occur during the formatting operation, the format function is canceled, the message

DMKFMT735E FATAL DASD I/O ERROR CSW = csw

is issued, and the next control statement is read.

Allocation Operation

In an allocation operation, disk space is assigned on the specified device in units of one cylinder for count-key-data or one page for FB-512. This disk space may be used as:

- Temporary space (TEMP)
- Permanent space (PERM)
- Directory space (DRCT)
- Temporary user space (TDSK)
- Paging space (PAGE)
- CP Dump space (DUMP)
- Override file space (OVRD).

The input parameters provide the information needed to update the allocation table. When the END allocation statement is processed:

- The allocation table is written. For count-key-data, this is the byte allocation map on cylinder 0, track 0, record 4 of the DASD device. For FB-512, this is the allocation extent map in blocks 3 and 4.
- The results of the allocation operation are displayed at the console.

The DASD device must already be formatted before an allocation operation can be performed.

Label-Only Operation

Count-Key-Data

In a label-only operation, a new volume is written on cylinder 0, track 0, record 3 of the DASD device. No label checking is done before the new label is written. The device must already be formatted before a label operation can be performed.

FB-512

In a label-only operation, a new volume label is written in the volume label block (block 1). A label-only operation can be done any time (the volume need not be formatted first).

Executing the Format Program

The sequence for executing the Format program is:

1. Ready the DASD device.
2. Ready the reader. The reader must contain the Format/Allocate program and may also contain control cards for the program.
3. IPL the reader.
4. If a console is not located at either address 009 or 01F, signal attention from the console so the Format program can establish the address of the console.
5. The program title is printed.
6. When there are no control cards in the reader, the program requests control statements by sending prompting messages to the console.
7. When control cards are in the reader, they are processed. The prompter messages are displayed with the response field updated from the control statements already entered through the card reader. The program requests additional input, which can be entered via the reader or console.
8. The program issues messages indicating the start or end of an operation.
9. An operation in progress may be canceled by signaling attention from the console. Execution resumes with the next operation.
10. The Format/Allocate program cancels an operation if a an unrecoverable DASD I/O error occurs. A message indicating the cause of the error is displayed.

Method of Operation

This section describes the execution of the disk format program and shows the processing associated with:

- Formatting DASD space
- Allocating DASD space
- Writing a volume label.

Figure 1 shows the relationship of the illustrations.

Figure 2 on page 8 describes the major functions of the Format/Allocate program.

Figure 3 on page 9 describes the format function for count-key-data.

Figure 4 on page 10 describes the allocate function for count-key-data.

Figure 5 on page 11 describes the format function for FB-512.

Figure 6 on page 12 describes the allocate function for FB-512.

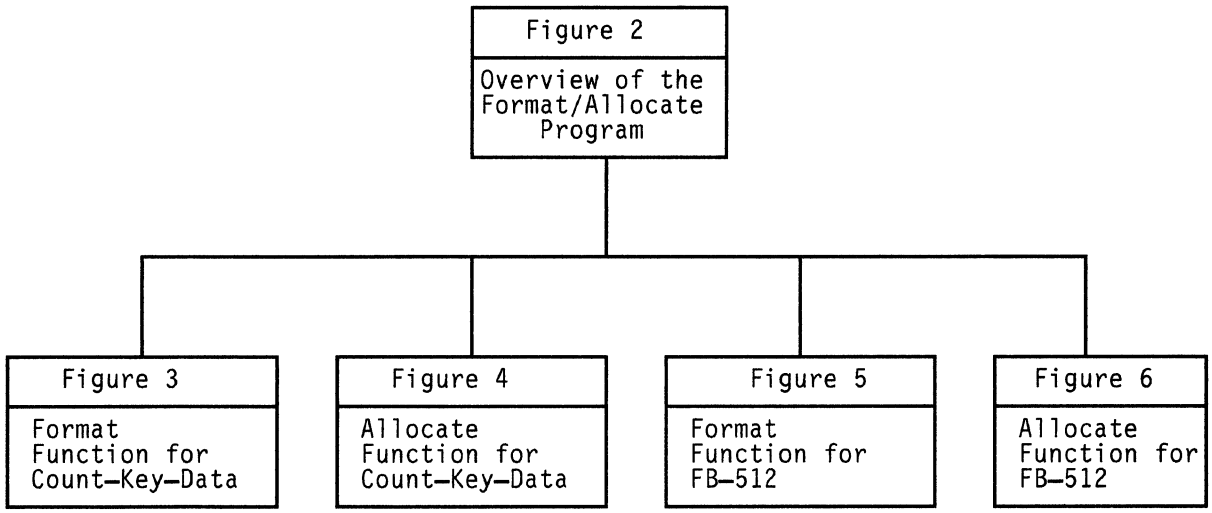
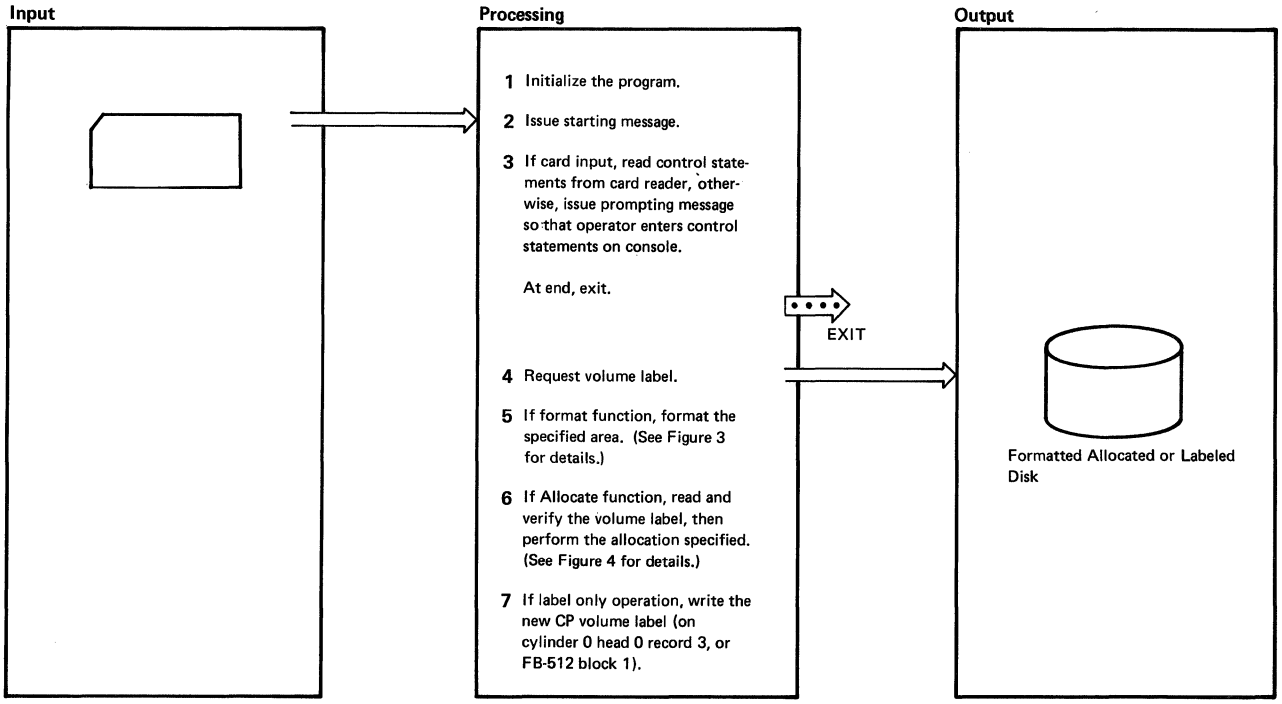


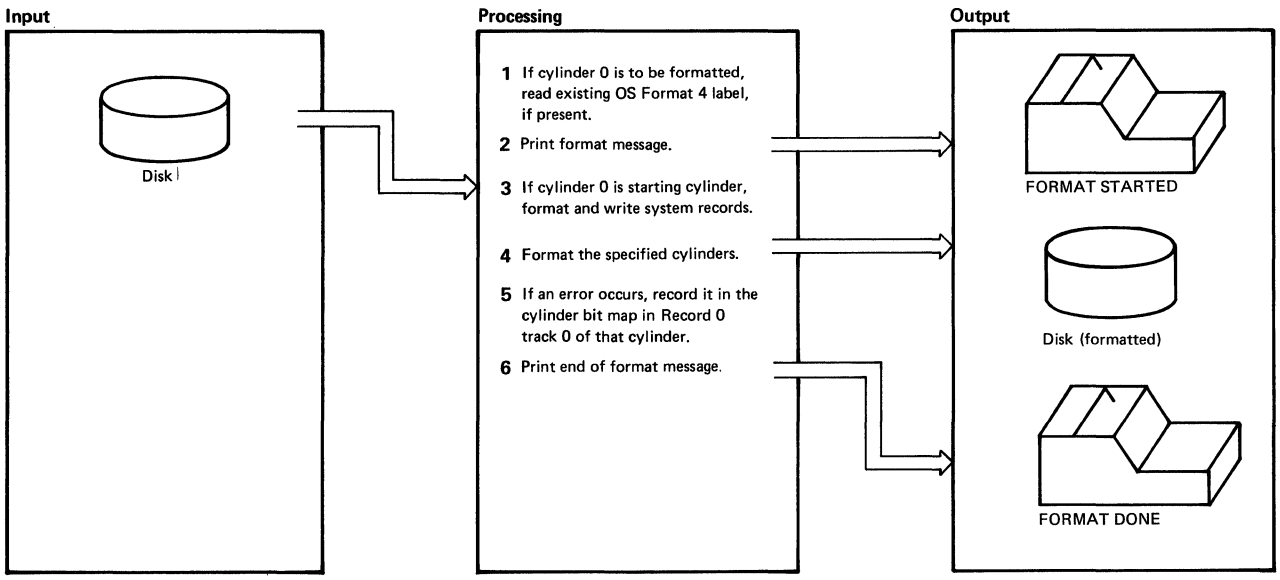
Figure 1. Overview of the Format/Allocate Program Figures

Format/Allocate



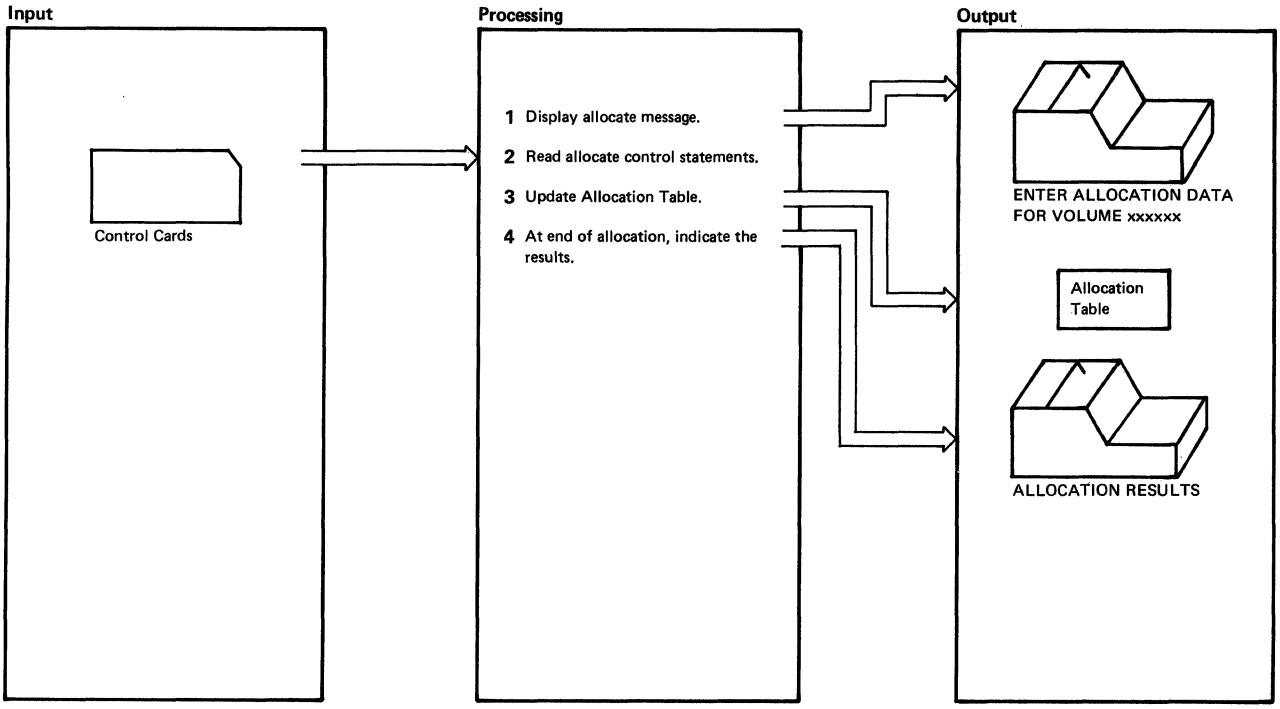
Notes		Module	Label	Ref
1	DMKFMT sets up registers 15, 11, 13, 8, and 12 as base registers, gets the IPL device address from the I/O old PSW, and stores it in IPLDEV. Next DMKFMT locates the consoles by testing 009 and 01F. If neither of these devices is available, it enters the wait state until an attention interruption is received from the console.	DMKFMT	DMKFMT	
2	The program title VM/370 FORMAT/ALLOCATE PROGRAM is displayed at the console.	DMKFMT	STMSG	
3	If the switch (CDSW2) contains X'FF', the reader enters the wait state until an I/O interrupt occurs. The CONSINT routine reads the control statements and the validate routine checks that they are valid.	DMKFMT	GETCARD CONSINT VALIDATE	
	The prompter messages are issued. If the control statements are entered through the card reader, the prompter messages include the response that was already specified in cards.		SELECT	
	The message ENTER FORMAT OR ALLOCATE prompts the operator. If the operator correctly enters FORMAT(F) or ALLOCATE(A), one of the following messages FORMAT FUNCTION SELECTED ALLOCATE FUNCTION SELECTED appears on the console. Otherwise, the prompter message is reissued. Then the message is reissued. Then, the message ENTER DEVICE ADDRESS (cuu): prompts the entering of the device address. If the device address entered is valid, the device type is requested. ENTER DEVICE TYPE: For count-key data, the high cylinder address, highest record, and device type are initialized depending on the device type entered.		DEVICEAD DEVICEAD DEVTYPE	
	For FB-512 devices, a "read device characteristics" CCW is performed and the highest block number is determined. From this, the highest page number is calculated.			
	If the device address entered is not available, the error message DMKFMT730E DEVICE rdev NOT OPERATIONAL OR NOT READY is issued and the request for a device is repeated.			
4	The message ENTER DEVICE LABEL: is displayed.	DMKFMT	LAB	
5	If the function being performed by the Format/Allocate program is the format operation, then, if cylinder 0 or page 0 is to be formatted, DMKFMT branches to FMT, otherwise, it branches to REGFORM1.	DMKFMT	LAB	
6	The volume label is read and verified by the LBLREC or FBALABRD CCW string, then DMKFMT branches to the ALLOCATE routine.	DMKFMT	LAB	
7	The CP volume label is written by the LABWRITE or FBAELIOW CCW string. Processing continues by reading the next control statement (see Step 3).	DMKFMT	LABONLY	

Figure 2. Overview of the Format/Allocate Program



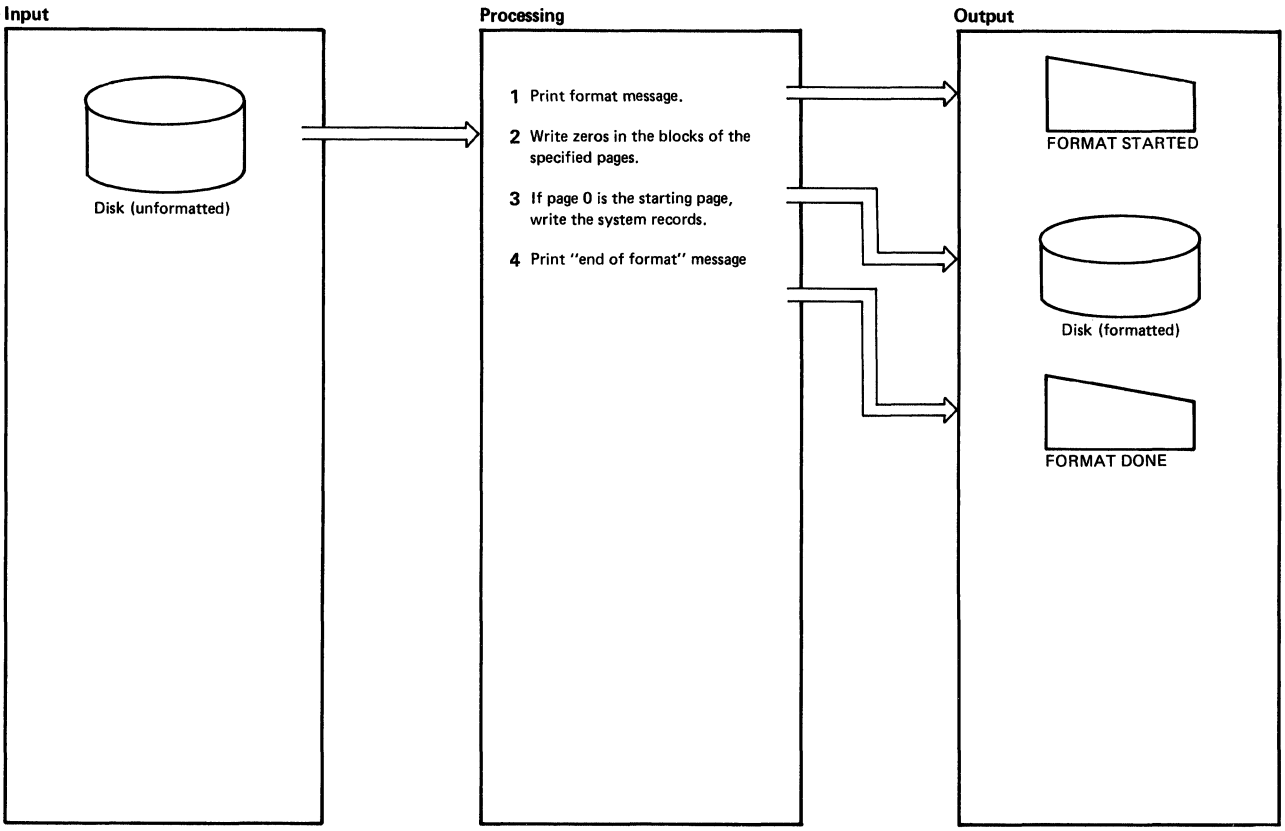
Notes		Module	Label	Ref
1	If cylinder 0 is to be formatted, any existing OS Format 4 label is read to preserve (for IBCDASDI) the CCHH address of the next unassigned alternate track and also the count of the remaining unassigned alternates. This data will be put in the new OS Format 4 label on track 0.	DMKFMT	FMT	
2	DMKFMT branches and links to the message writing (WMSG) routine to display FORMAT STARTED Then it updates the I/O new PSW so that the IONT routine executes when an I/O interrupt occurs.	DMKFMT	REGFORM1	
3	If cylinder 0 is the starting cylinder, the format program formats cylinder 0 by setting up the CCWs appropriate to the device type and then branching to the STIO routine to perform the I/O operation. Once cylinder 0 is formatted, system records are written on it. Then branch to the CHECK0 routine is set to NOP so that CHECK0 is executed only once. The records written on cylinder 0 are **Record** **Description** 0 Page bit map 1 IPL record 2 Checkpoint record 3 Vol1 label 4 Allocation bit map 5 Format 4 label 6 Format 5 label F3 Page size filler F4 Filler record for 2314/2319	DMKFMT	STORE CHECK0	
4	The appropriate device type CCWs are set up by the Format program. Page size records are written and verified by the STIO routine. Control returns to the RESUMP routine if no error occurs. The RESUMP routine updates the record numbers and the STIO routine again writes and verifies the record. This loop continues until the last cylinder specified is completely formatted.	DMKFMT	STORE STIO RESUMP	
5	If an error occurs in the STIO routine, control is transferred to the IOINT routine. The error is retried up to 9 times before the message DMKFMT736E IO ERROR rdev {CCHHR=cchhr|BLOCK=nnnnnn} SENSE=sense is displayed. The Page bit map is updated to indicate a bad surface. The errors that cause the Format function to terminate are: seek error error in writing or reading the home address error writing or reading record 0 error setting file mask error in reading count-key-data The message DMKFMT735E FATAL DASD IO ERROR is displayed and control returns to the GETCARD routine.	DMKFMT	IOINT READER06	
6	DMKFMT displays the message FORMAT DONE to indicate that the specified cylinders are formatted, and then summarizes the errors with the message xxxPAGE RECORDS FLAGGED	DMKFMT	CLEANUP	

Figure 3. Format Function for Count-Key-Data



Notes		Module	Label	Ref
1	The messages ENTER ALLOCATION DATA FOR VOLUME xxxxxx type cyl cyl .... are displayed.	DMKFMT	ALLOCATE	
2	If the Allocate control statements are entered via a card reader, the switch (CDSW2) contains X'FF'. Control is transferred to the GETCARD routine which reacts the cards. The CONSINT and VALIDATE routines verify the control statements and allocate processing resumes at the label REREAD. There is a branch and link to the RMSG routine to read from the console. The console read is not performed in this case because CDSW is X'FF'. If the allocate control statements are entered via the console, the switch (CDSW2) contains X'FF'. The control statements are read from the console by branching and linking to the RMSG routine.	DMKFMT	GETCARD CONSINT VALIDATE REREAD REREAD RMSG	
3	The address of the cylinder byte map is located into register 9. The total number of cylinders specified is loaded into register 8. The cylinder byte map is updated for each of the specified cylinders according to the type indicated in the control statement. **Control Statement** / **Indication in Cylinder Byte Map** TEMP X'00' PERM X'01' TDSK X'02' DRCT X'04' PAGE X'08' DUMP X'10' OVRD X'14' The map is printed after the END statement is processed.	DMKFMT	AOKALL INDIC	
4	The message ALLOCATION RESULTS followed by the type corresponding to the allocated cylinders is displayed. Finally, the message DEVICE xxx VOLUME xxxxxx ALLOCATION ENDED is displayed.	DMKFMT	FINI	

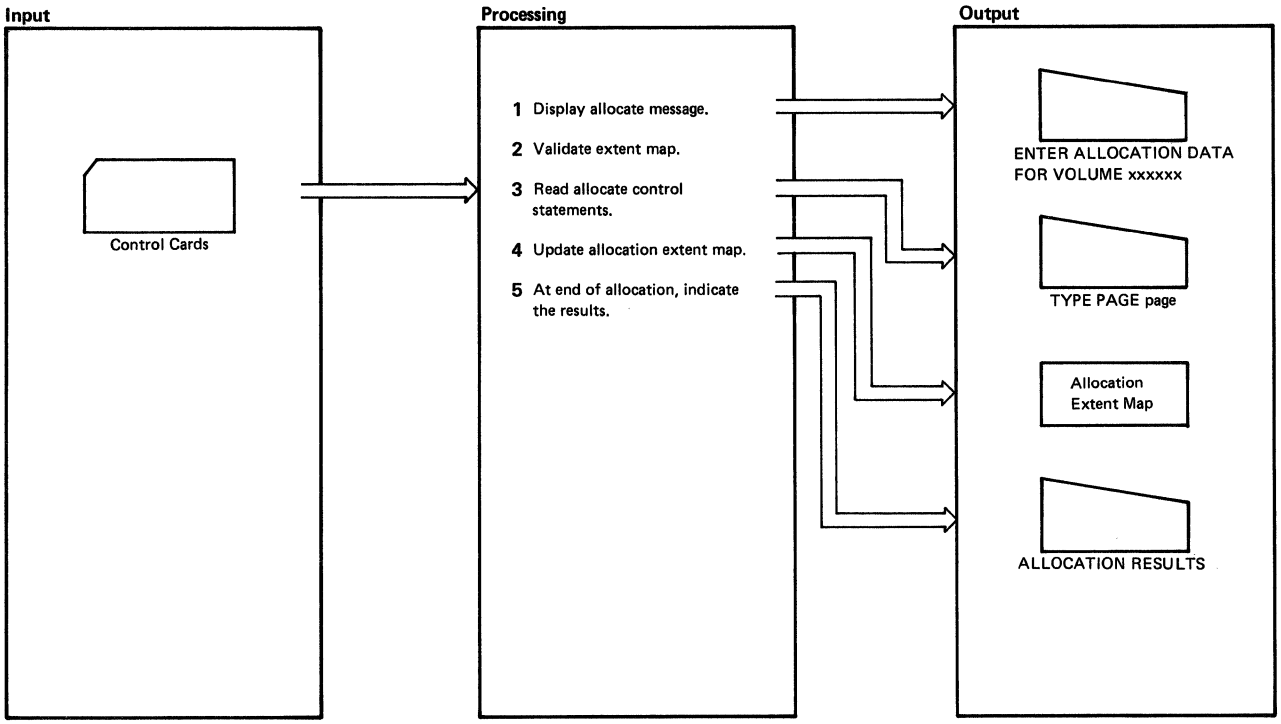
Figure 4. Allocate Function for Count-Key-Data



Notes		Module	Label	Ref
1	DMKFMT branches and links to the message writing (WMSG) routine to display FORMAT STARTED Then it updates the I/O PSW so that the IOINT routine executes when an I/O interrupt occurs.	DMKFMT DMKFMT	REGFORM1 FORMFBA	
2	Define extent data is initialized to map the entire volume. The FORMCCW string is used to write zeros in the blocks of the specified pages. Each call to the STIO routine writes a track's worth of blocks. The RESUMFBA routine increases to the next block number and the STIO routine again writes and verifies the blocks. This loop continues until the last block of the last page has been written.	DMKFMT	FORMFBA NEXTCAG FORMIO RESUMFBA	
3	If page 0 was specified as the starting page, write the system data in the first 16 blocks. The blocks written are: **Block** **Description** 0 IPL record 1 Vol1 label 2 VTOC 3-4 Allocation map 5-12 DMKCKP program 13-15 Reserved - all zeros	DMKFMT	FORMEND	
4	DMKFMT displays the message FORMAT DONE to indicate that the specified cylinders are formatted, and then summarizes the errors with the message xxxPAGE RECORDS FLAGGED	DMKFMT	CLEANUP	

Figure 5. Format Function for FB-512

Format/Allocate



Notes		Module	Label	Ref
1	Display the message ENTER ALLOCATION DATA FOR VOLUME xxxxxx	DMKFMT	ALLOCATE	
2	Validate the extent map. If it is empty, go to Step 3. If there is data already in the map, verify that the entries are numerically ascending and that a X'FF' marks the end of the map. If the validity test fails, set a flag (BADFLAG) and go to Step 3.	DMKFMT	VALOOP	
3	Display the message TYPE PAGE page if the allocate control statements are entered via a card reader, the switch (CDSW2 contains X'FF'. Control is transferred to the GETCARD routine, which reads the cards. The CONSINT and VALIDATE routines verify the control statements and allocate processing resumes at the label REREAD. There is a branch and link to the RMSG routine to read from the console. The console read is not performed in this case because CDSW2 is X'FF'.	DMKFMT	INITMAP CONSINT VALIDATE REREAD	
	If the allocate control statements are entered via the console, the switch (CDSW2) contains X'00'. The control statements are read from the console by branching and linking to the RMSG routine.		REREAD RMSG	
4	Each input is checked for validity. A dummy entry is created for the extent map and is used to scan the map looking for the correct insertion point. If the new entry overlays existing entries in any manner, the extent map is reconstructed, to fit in the new entry. The 'type' byte of the 12-byte extent map is set based on the input control statement: **Control Statement / Type Byte** TEMP X'00' PERM X'01' TDSK X'02' DRCT X'04' PAGE X'08' DUMP X'10' OVRD X'14'	DMKFMT	ALOCFBA	
5	The message ALLOCATION RESULTS is displayed following by the type and corresponding start and end page numbers. Then the message DEVICE xxx VOLUME xxxxxx ALLOCATION ENDED is displayed.	DMKFMT	PRINTALL	

Figure 6. Allocate Function for FB-512

Program Organization

DMKFMT

A standalone program that formats, allocates, and labels all (or part) of 2314, 2319, 3330, 3340, 3350 Series, 3375, 3380, FB-512 Series, and 2305 Series Direct Access Storage Devices for VM/SP use.

Entry Point

DMKFMT

Routines Called

None

Register Use

Reg	Use
R0-7	Scratch
R8	5th base register
R9-10	Scratch
R11	3rd base register
R12	2nd base register
R13	4th base register
R14	Scratch
R14	Linkage register
R15	1st base register

Directory

Following is an alphabetical list of the major labels in the Format/Allocate program. The associated method of operation diagram and a brief description of the function performed at the point in the program indicated by each label are included in the list.

Label	Figure	Description
ALLOCATE	4	Performs the allocate function of the Format program (count-key-data)
ALOCFBA	6	Performs the allocate function of the format program (FB-512)
ALTTRACK		Performs alternate track recovery for 3340/3344
AOKALL	4	Locates the cylinder byte map
CHECK0	3	Writes system records on cylinder 0
CLEANUP	3	Summarizes the errors encountered while formatting the disk
CONSINT	2, 4	Processes console interrupts
DEVICEAD	2	Displays the prompter message requesting the device address
DEVTYPE	2	Displays the prompter message requesting the device type
DMKFMT	2	Initializes the Format program
ERRECOV		Performs DASD error recovery
FATAL		Displays the termination message and reads the next control statement
FINI	4	Displays the cylinders just allocated with the type of allocation
FMT	3	Initializes cylinder 0 for formatting by first reading any existing OS Format 4 label
FORMAL		Displays the starting cylinder or label message
FORMFBA	5	The main FB-512 formatting routine
GETCARD	2, 4	The main control routine. It reads control statements from the reader or transfers control to the SELECT routine to issue prompter messages.
GRAPHID		Handles input and output operations for display terminals
INDIC	4	Updates the cylinder byte map to reflect the type of allocation for each cylinder
IOINT	3	Handles I/O interrupts and retries errors
LAB	2	Displays the prompter message requesting the device label
LABELIOR		Reads and verifies the volume label
LABONLY	2	Rewrites the volume label (record 3) and nothing else
MCRTN		Processes machine checks
NEXT		Displays end of cylinder message
PRINTALL		Displays the allocation table on the terminal
READER06	3	Updates the page bit map to indicate a bad surface
REGFORM1	3	Initializes the format function when cylinder 0 is not included
REREAD	4	Reads control statements from the console for the allocate function
RESUMFBA	5	Updates the block number during the format operation (FB-512)
RESUMP	3	Updates the record number during the format operation (count-data-key)
RMSG	4	Reads from the typewriter terminals
SELECT	2	Prompts the operator to enter the appropriate control statement
SENSIT		Gets sense information
SENSIT2		Displays the sense information
STIO	3	Writes and verifies page size records during format operation
STMSG	2	Displays the program title

Label	Figure	Description
STORE	3	Sets up CCW string to format cylinder 0
VALIDATE	2, 4	Checks control statements entered through a card reader for accuracy
WMSG		Displays messages on the terminal
XBIN		Converts hexadecimal numbers to binary

Data Areas

This section contains descriptions of the layout of the DASD records and the DASD record formats for:

- 2305 Models 1 and 2
- 2314/2319 devices
- 3330 series
- 3340 series
- 3350 series
- 3375
- 3380
- FB-512 series.

Record Layout

2305 Models 1 and 2

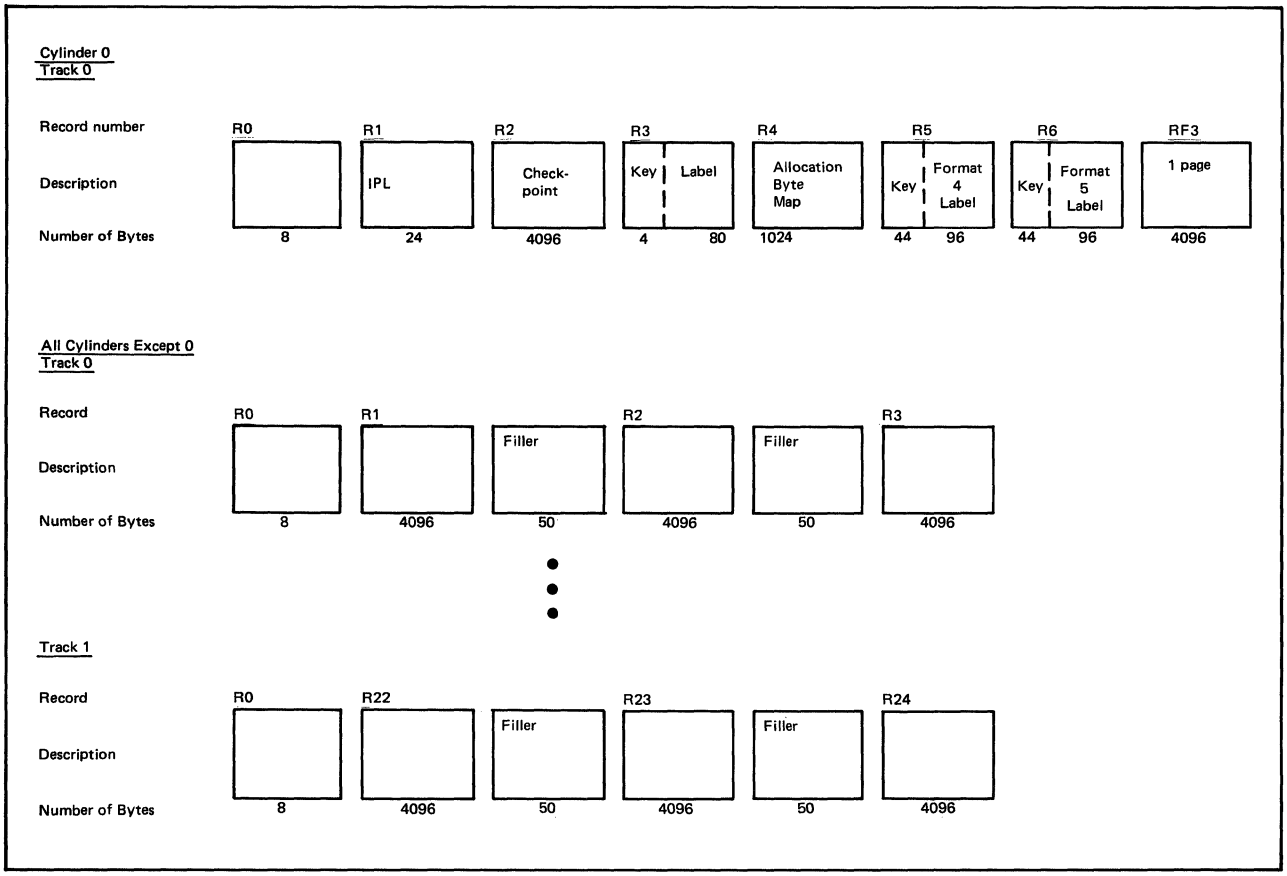


Figure 7. 2305 Models 1 and 2 Record Layout

2314/2319

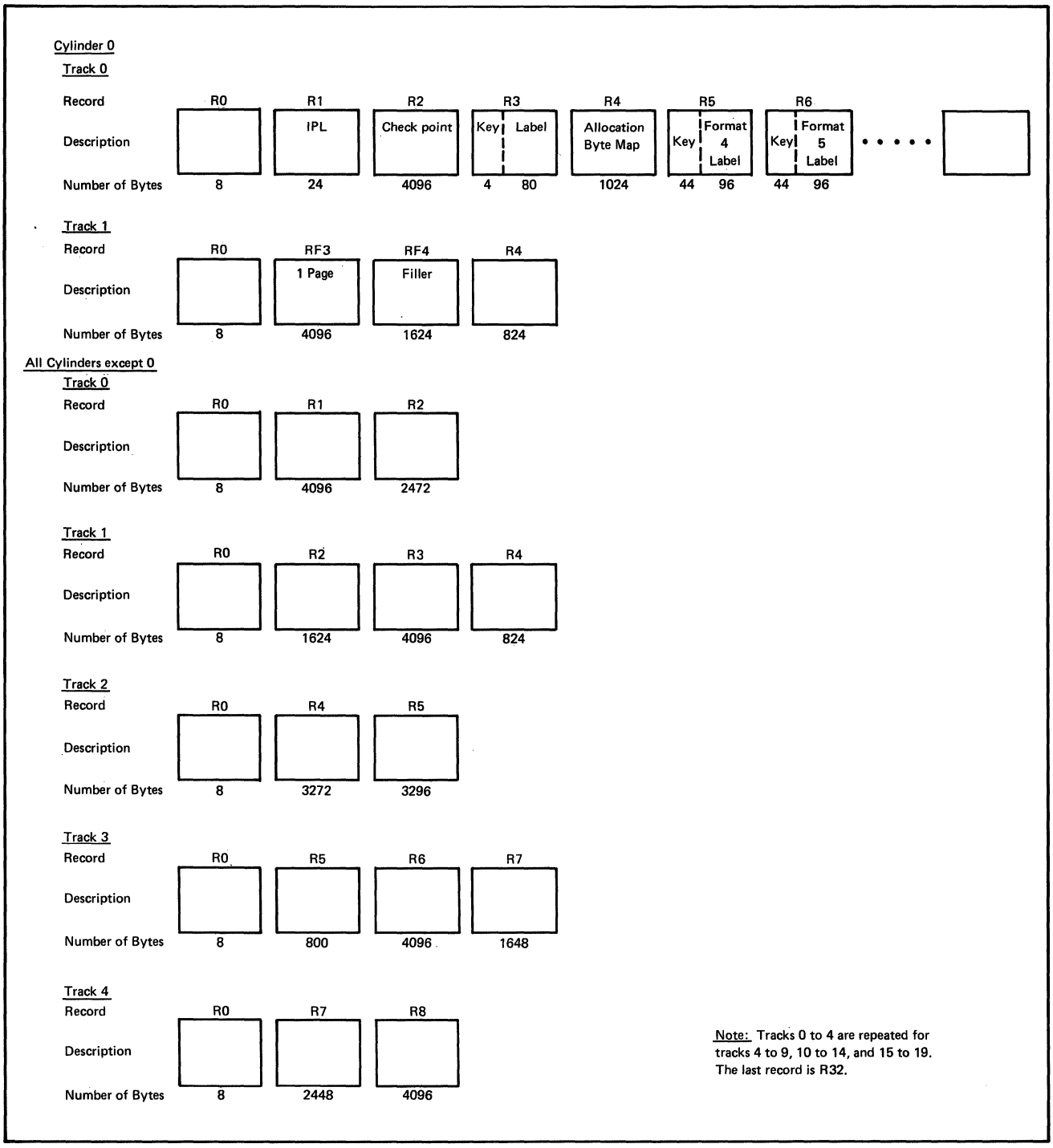


Figure 8. 2314/2319 Record Layout

3330 Series

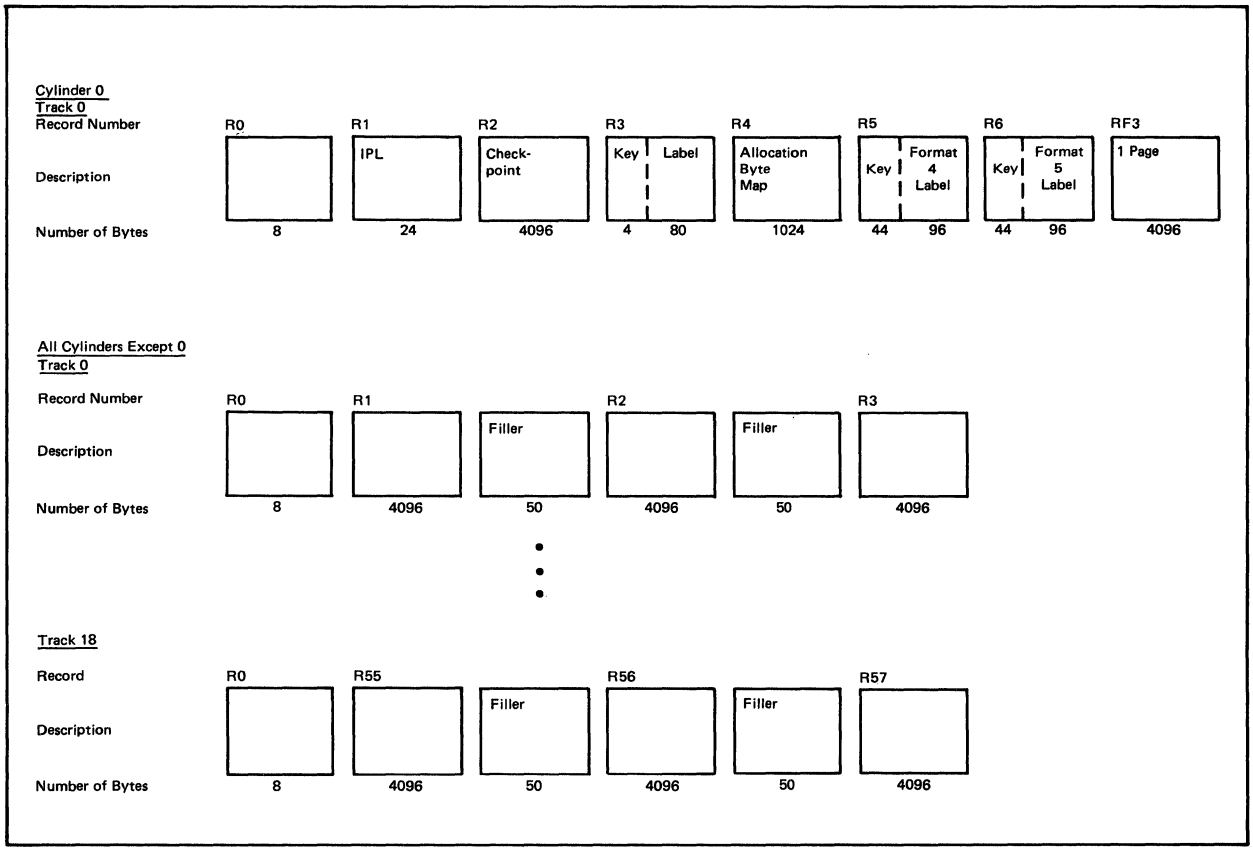


Figure 9. 3330 Series Record Layout

3340

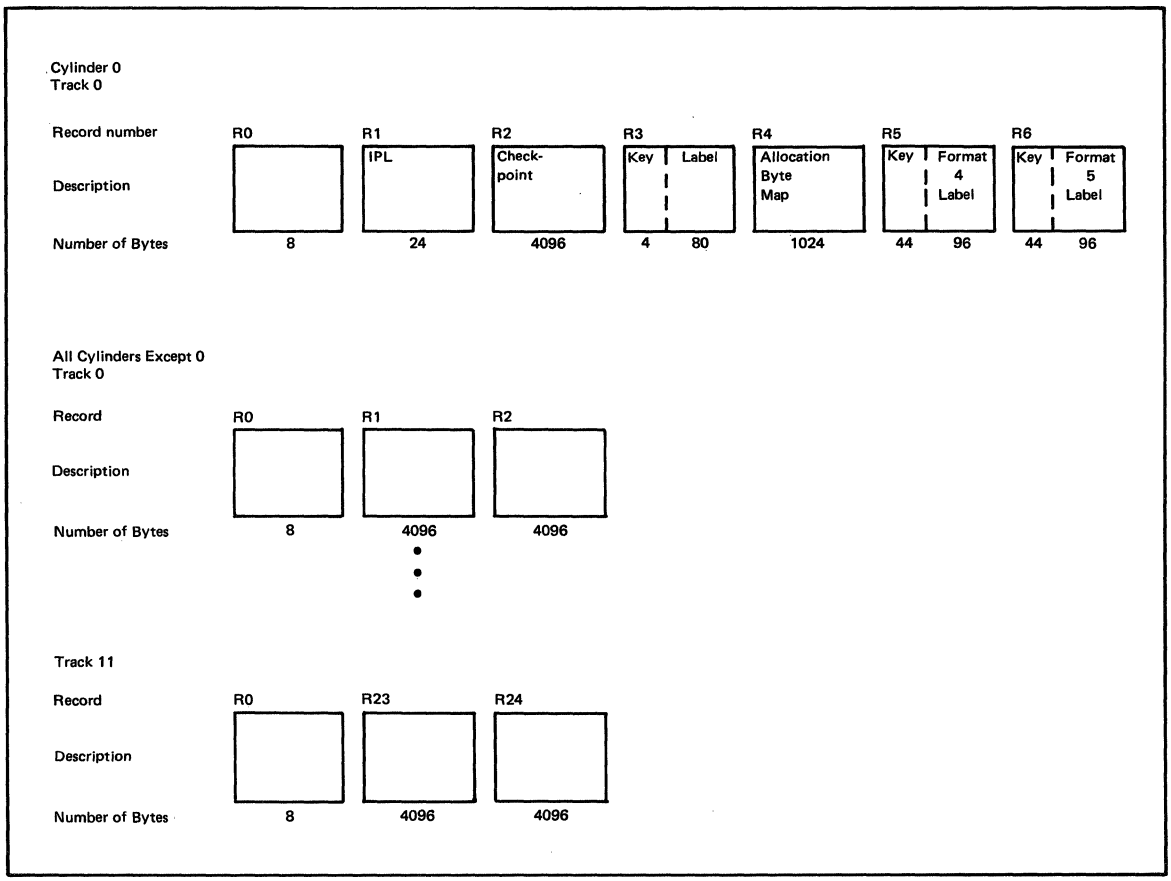


Figure 10. 3340 Record Layout

3350

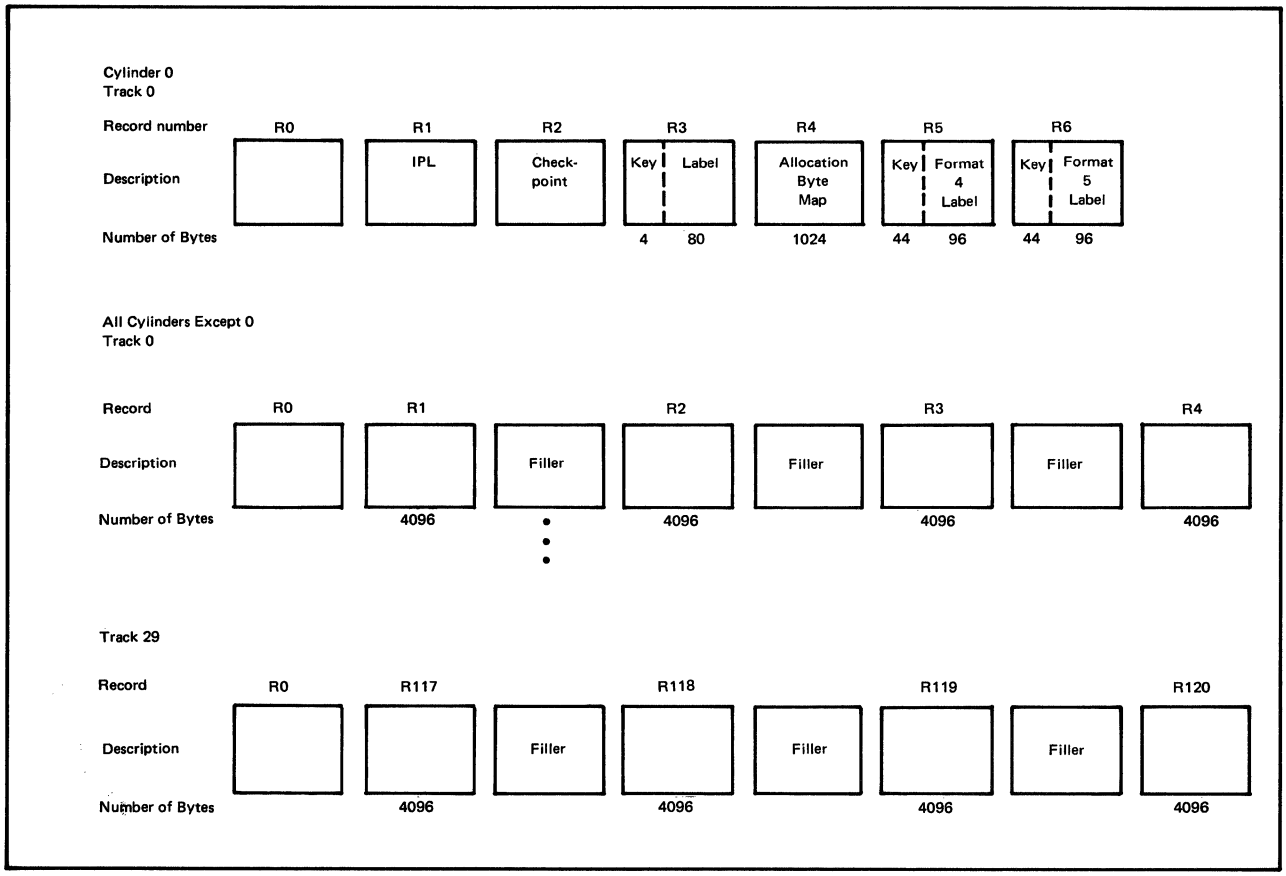


Figure 11. 3350 Record Layout

3375

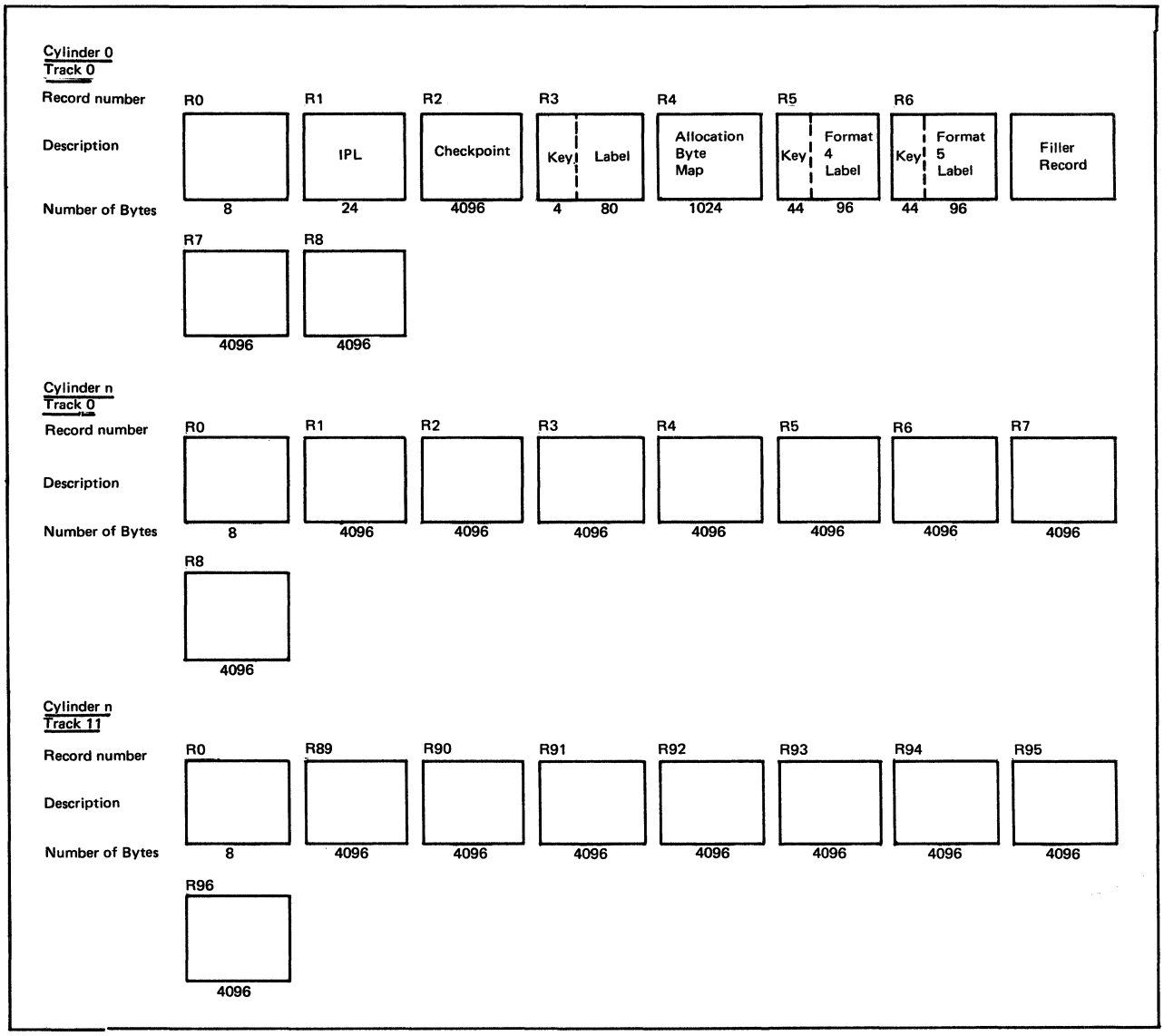


Figure 12. 3375 Record Layout

3380

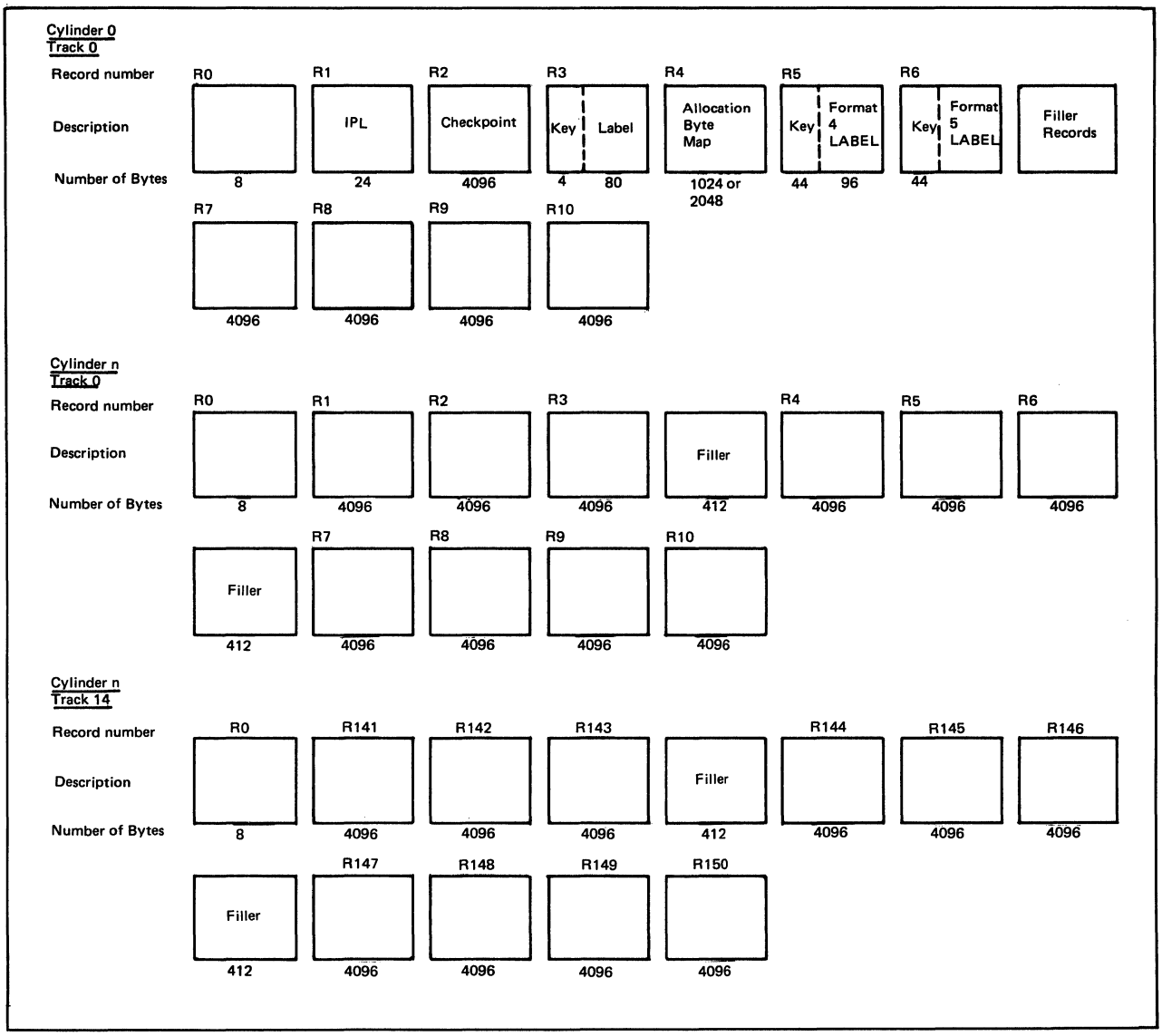
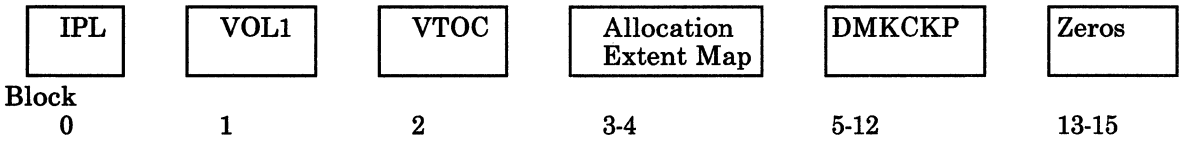


Figure 13. 3380 Record Layout

FB-512 Devices

FB-512 Data Layout and Content (each block is 512 bytes)



Blocks 16 to the end of the volume contain CP pages.

Figure 14. FB-512 Series Record Layout

Record and Block Formats

The following record descriptions pertain to the records that are in cylinder 0 unless another cylinder is identified.

Record 0

Record 0 is the standard 8-byte data area.

For all cylinders except 0, this record contains zeros.

For 2305, 2314/2319, 3330, 3340, 3380, and 3350 in compatibility mode cylinder 0 record 0 contains:

```
E0 00 00 00 00 00 00 00
```

For 3350 in native mode cylinder 0 record 0 contains:

```
F0 00 00 00 00 00 00 00
```

Record 1

Record 1 contains a 24 byte IPL record. It puts system into wait state if storage device is loaded (via IPL function). The 24 bytes are:

```
00 02 00 00 00 00 00 00 03 00 00 00 20 00 00 00 00 00 00 00
00 00 00 00
```

Record 2

Record 2 contains a 4096 byte checkpoint record. This is the checkpoint program that is loaded at VM/SP IPL time to retrieve and save control information for a warm start.

Record 3

Record 3 contains a 4-byte key and an 80-byte record. The 80-byte record contains:

Disp	Description
0	Key
4	Label
10	F0
11	Pointer to VTOC
16	00
21	40
41	00
46	CP370
51	40
52	Pointer to user directory
56	40

Record 4

Record 4 is a 1024-byte allocation byte map. It is used to identify cylinder usage. Each byte corresponds to one cylinder; the value of the byte indicates the type of usage for the cylinder.

Value	Use
00	Temporary
01	Permanent
02	T-disk
04	Directory
08	Paging
10	CP dump
14	OVRD
FF	This identifies the cylinder beyond the last cylinder that can be allocated.

Record 4 is an 824-byte record that can be on track 1 of 2314/2319 devices only. The first segment of Record 4 is used for paging.

Record 5

Record 5 contains a 44-byte key and a 96-byte data area. The 96-byte data area contains a format 4 DSCB type label - used to be compatible with other systems.

Record 6

Record 6 contains a 44-byte key and a 96-byte data area. The 96-byte data area contains a format 5 DSCB type label - used to be compatible with other systems.

Record F3

Record F3 is a 4096-byte record that can be on track 0 or 1 of 2314/2319 devices only. It is reserved for IBM use, and is referred to as a filler record.

Record F4

Record F4 is a 1624-byte record that can be on track 1 of 2314/2319 devices only. It is used to align Record 4 on the track.

Block 0

Block 0 contains a 24-byte IPL record. The remainder of the block contains zeros. It puts system into wait state if storage device is loaded (via IPL function). The 24 bytes are:

```
00 02 00 00 00 00 00 00 03 00 00 00 20 00 00 00 00 00 00 00
00 00 00 00
```

Block 1

Block 1 contains an 80-byte volume label. The remainder of the block contains zeros. The 80-byte record contains:

Disp	Description
0	VOL1
4	Label
10	F0
11	Pointer to VTOC. Contains 00 00 00 02
16	00
21	CI size. Contains 00 00 02 00
25	Blocks per CI. Contains 00 00 00 01
29	Labels per CI. Contains 00 00 00 03
33	4040
41	00
46	CP370
51	40
52	Pointer to user directory
56	40

Block 2

Block 2 contains the Volume Table of Contents. It is a Format-4 and Format-5 DSCB. This block contains:

Disp	Description
0	0404
44	F4
45	0000
59	01
61	00000000
107	VTOC start address Contains 00000002
111	VTOC end address Contains 00000003
115	00000000000000
140	0505
184	F5
186	00000000

Blocks 3-4

Blocks 3-4 contain a 1024-byte allocation extent map. It is used to identify the use of pages on the device. It is made up of 12-byte entries.

Each entry represents an extent of pages and the use of the extent. Each 12-byte entry contains:

Disp	Description
0	Type

The type field can contain:

Value	Use
00	TEMP
01	PERM
02	TDSK
04	DRCT
08	PAGE
10	DUMP
14	OVRD
FF	Marks the end of the table.

Disp	Description
2	Unused
4	Start
8	End

Start and End are page numbers denoting the range. The entries are sorted from low to high.

Blocks 5-12

Blocks 5-12 contain the 4096-byte checkpoint record. This is the first page of the checkpoint program (DMKCKP) that gets control at VM/SP IPL time to retrieve and save control information for a warm start.

Blocks 13-15

Reserved for system use - contains zeros.

Diagnostic Aids

Following is a list of the messages issued by the Format/Allocate program. The label of the message, the label of the routine issuing the message and the associated method of operation diagram are included in the list.

Message Code	Label of Message	Issuing Routine	Figure	Message Text
DMKFMT536I	DRCTFAIL	CSWMSS		rdev REPORTS DISABLED INTERFACE; FAULT CODE = code; NOTIFY CE
DMKFMT730E	WR1	DEVTYPE	2	DEV rdev NOT OPERATIONAL OR NOT READY
DMKFMT732E	MCMSG	MCRTN		MACHINE CHECK
DMKFMT733E	WRONG	LABELBAD		VOLID READ IS volid1 NOT volid2
DMKFMT734E	TYPERR	VALIDATE ALLOCATE		TYPE OR {CYL|PAG} INVALID
DMKFMT735E	FATLMSG	FATAL	3	FATAL DASD I/O ERROR CSW = csw
DMKFMT736E	IOERR	DEVICEAD SENSIT2	3	IO ERROR rdev {CCHHR = cchhr|BLOCK = nnnnnn} SENSE = sense
DMKFMT737E	BAD	BADINPUT		INVALID OPERAND
DMKFMT738A	IPLERROR	DEVICEAD		DEVICE rdev INTERVENTION REQUIRED
DMKFMT739E	MSGATRK	ALTTRACK		FLAGGED PRIMARY TRACK HAS NO ALTERNATE ASSIGNED; I/O ERROR FOLLOWS
DMKFMT740E	MSG35MB	DEVTYPE	2	PACK MOUNTED IS 3340-35, NOT 3340-70. MOUNT ANOTHER OR RESPECIFY
DMKFMT741E				DEVICE rdev IS devtype1 NOT devtype2 AS SPECIFIED. RESPECIFY OR NOTIFY SYSTEM SUPPORT
DMKFMT742E				ALLOCATION FUNCTION NOT ALLOWED - FORMAT OF VOLUME IS A PREREQUISITE
DMKFMT756E	PCMSG	PRCHK ERRORMSG		PROGRAM CHECK PSW = psw
	TITLE	STMSG	2	VM/370 FORMAT/ALLOCATE PROGRAM RELEASE n
	FORA	SELECT	2	ENTER FORMAT OR ALLOCATE:
	FMTMSG	SELECT	2	FORMAT FUNCTION SELECTED
	ALLOCMSG	SELECT	2	ALLOCATE FUNCTION SELECTED
	ADDRESS	DEVICEAD	2	ENTER DEVICE ADDRESS (CCU):
	TYPMSG	DEVTYPE	2	ENTER DEVICE TYPE:
	DATAMSG	ALLOCATE	4	ENTER ALLOCATION DATA FOR VOLUME xxxxxx
	ALMSG	ALLOCATE	4	TYPE CYL CYL
	ALMSG1	ALLOCATE	4	
	ALLEND	FINI	4	DEVICE xxx VOLUME xxxxxx ALLOCATION ENDED

Message Code	Label of Message	Issuing Routine	Figure	Message Text
	STCYL	FORMALL		ENTER START CYLINDER (xxx) OR "LABEL":
	ENDCYL	NEXT		ENTER END CYLINDER (xxx):
	PROGFOR	REGFORM	3	FORMAT STARTED
	RDLAB	LAB	2	ENTER DEVICE LABEL:
	ENDFOR	CLEANUP	3	FORMAT DONE
	PAGE	CLEANUP	3	xxx PAGE RECORDS FLAGGED
	RESULTS	FINI	4	ALLOCATION RESULTS
	MAP	PRINTALL		TEMP 000 000
	LABELCHK	LABONLY		LABEL IS NOW xxxxxx
	STPAGE	STRTPAG	2	ENTER START NUMBER OR "LABEL":
	ENPAGE	ENDPAG	2	ENTER END PAGE NUMBER:
	ALPMSG	INITMAP	5	TYPE PAGE page
	MAPFULL	COMPRESS	5	NUMBER OF EXTENTS EXCEEDS MAXIMUM - RESPECIFY
	PAGEXC	ALOCFBA	6	HIGHEST ALLOCATABLE PAGE IS xxxxxx - RESPECIFY
	PAG2LO	ALOCFBA	6	LOWEST ALLOCATABLE PAGE IS PAGE 2 - RESPECIFY
	PAGERR	ALOCFBA	6	PAGE NUMBER INVALID - RESPECIFY
	FBAMAP	FBAPRALL	6	TYPE xxxxxx xxxxxx

Index

Chapter 2. Directory Program

Directory Program

Introduction

The DMKDIR program builds the VM/SP directory on a volume previously formatted by the Format/Allocate program, using space that was previously allocated for use as directory space.

Under the control of the VM/SP system, the new directory is dynamically swapped and placed in use provided the directory has been created without errors, on a volume in the system-owned list, and provided the user class is A, B, or C.

The new directory can be built so that it does not overlay an existing directory. To do this, allocate enough space for two directories or allocate space for a new directory each time the directory is created.

The directory program can be run standalone or under the control of CMS. The CMS DIRECT command invokes the directory program under CMS.

Method of Operation

This section describes the operation of the VM/SP Directory program. Figure 15 shows the relationship of the Method of Operation figures.

Figure 16 describes the major functions of the Directory program.

Figures 17, 18, and 19, describe the control statement processing and the resulting action.

Figure 20 shows the functions performed before the program terminates.

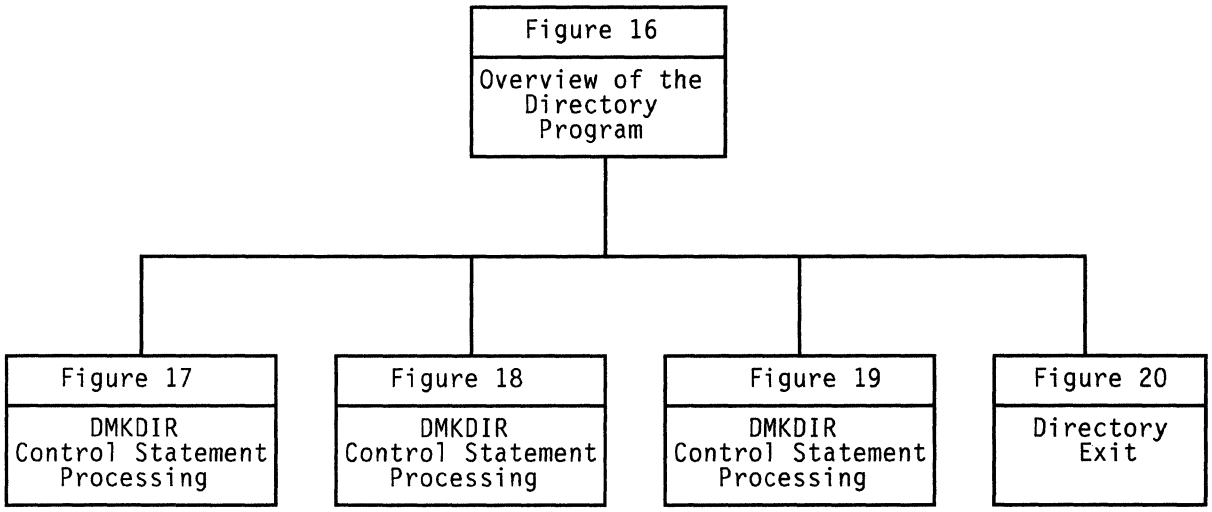
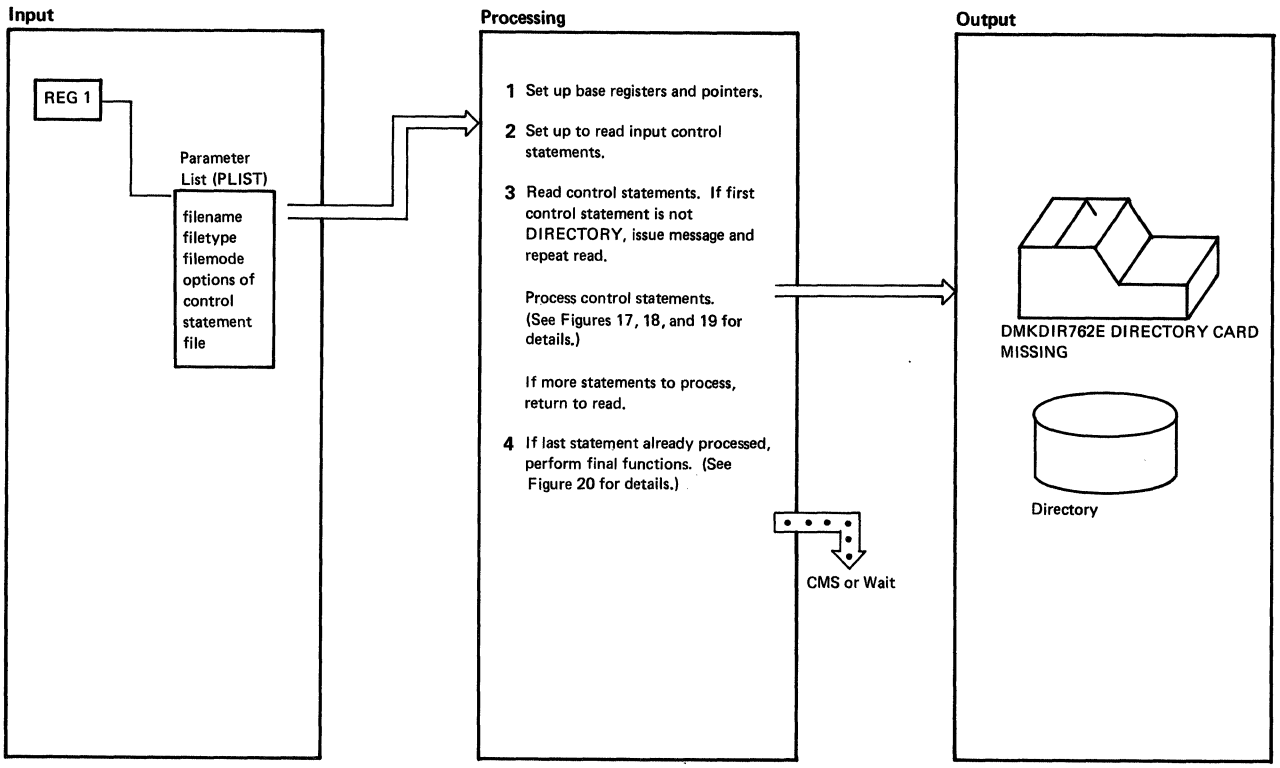


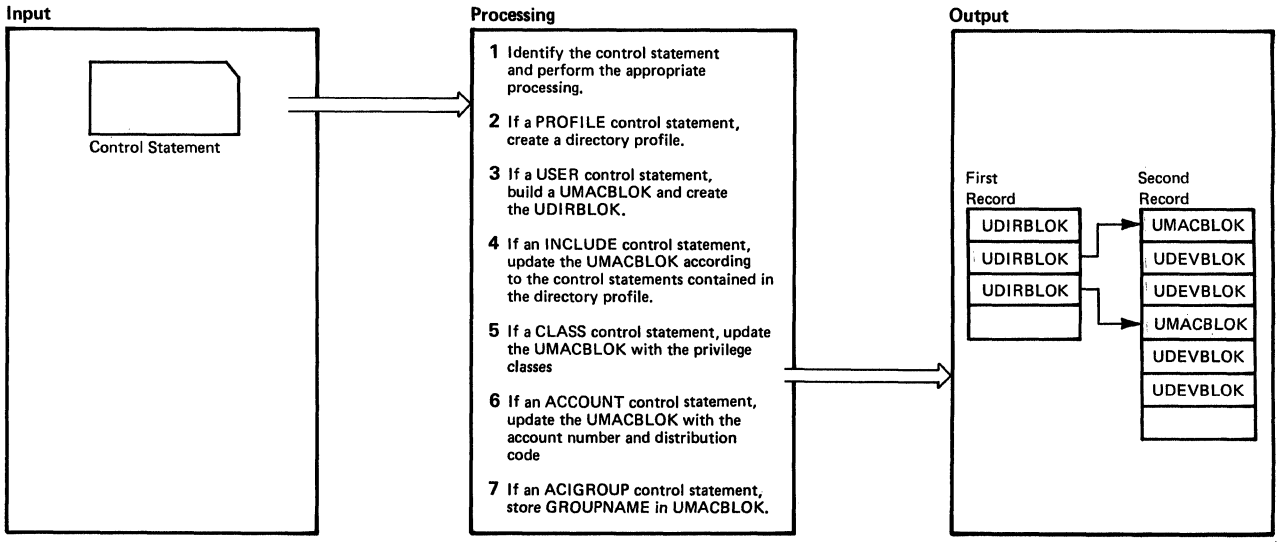
Figure 15. Key to the Directory Program Method of Operation Figures

Directory Program



Notes		Module	Label	Ref
1	DMKDIR sets up registers 12, 13, and 9 as base registers and sets up pointers to the first UDEVBLOK and the allocation record buffer.	DMKDIR	DMKDIRT	
2	If running standalone, the header line is printed:	DMKDIR	MSGRET	
	VM/370 USER DIRECTORY CREATION PROGRAM RELEASE n		MSG02A	
	ENTER CARD READER DEVICE ADDRESS AND OPTIONS			
	The program then reads a response from the console. A read is issued to the card reader indicated (if any). If the operator enters a null line in response to the message, the IPL device is used as the input card reader. If the EDIT option is specified, DIRFLAG is set to X'20'.		DEFAUL13 STOREADD	
	If running under CMS, set the P-list containing the filename, filetype and filemode of the file containing the directory control cards. If EDIT is specified, the DIRFLAG is set to X'20'.		CMS1 EDITTEST	
	The STATE macro is issued to see if the control statement file exists. If the file is not found, the messages DMKDIR763E INVALID FILE - NAME OR FILE NOT FOUND EOJ DIRECTORY NOT UPDATED are displayed and control returns to CMS.		STATE TERM	
3	The STATE macro is issued to see if the restricted password list file exists. If so, the FSREAD macro is issued to read the file into a buffer. If it does not exist, message DMK750W is issued, and processing continues.	DMKDIR	STATE	
4	Control statements are read via SVC 202 when the Directory program is run under the control of CMS. When the Directory program runs standalone, the read function is performed either by the GRAPHID routine (if the console is a display device) or by the STARTIO routine in all other cases. The READ routine scans the control statement and branches to the appropriate processing routine. After processing each control statement and executing the associated routine, control returns to READ to process the next control statement.		READ GRAPHID STARTIO	
4	When the last statement is read and processed, the READ routine branches and links to the EXIT routine.		EXIT	

Figure 16. Overview of the Directory Program

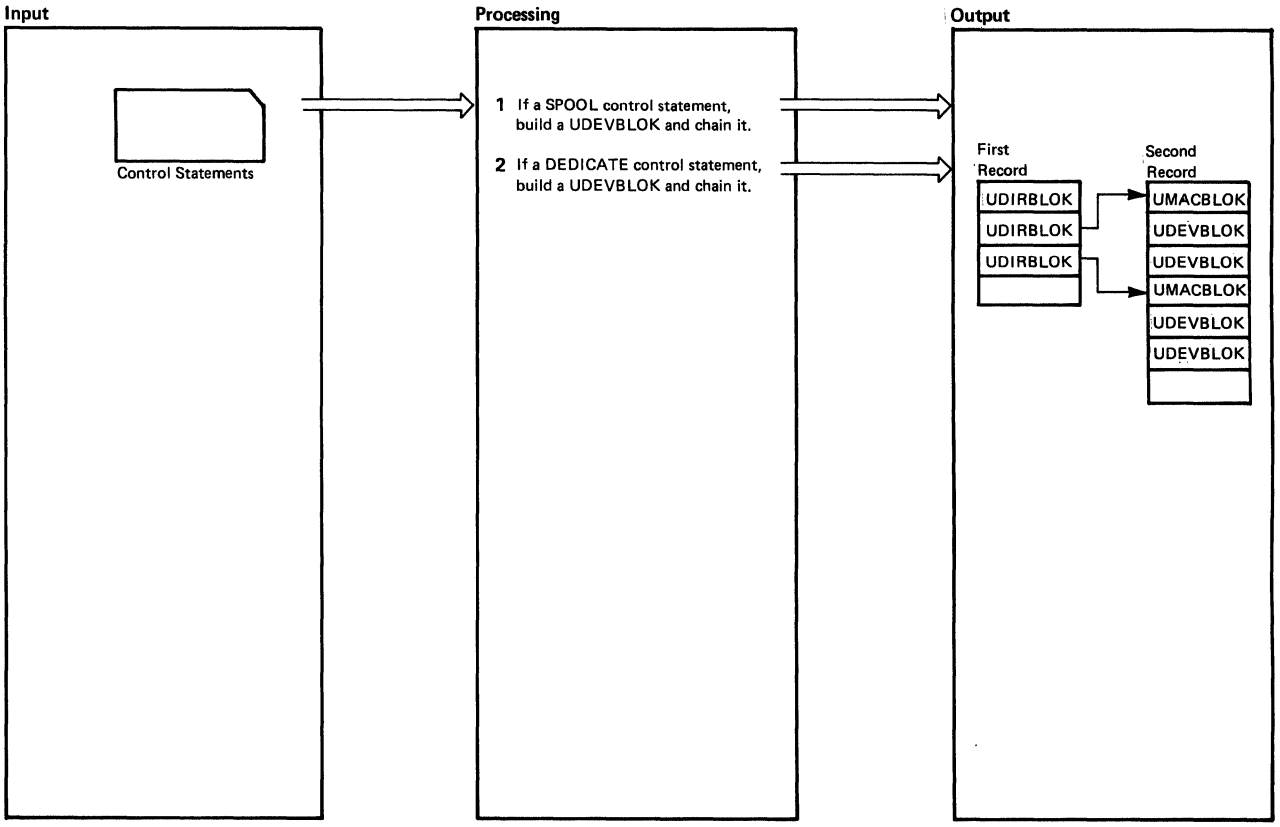


Notes		Module	Label	Ref
1	The READ routine branches and links to the SCANNAME routine with register 4 pointing to TABLE1. TABLE1 is searched for a keyword matching the control statement name and control is passed to the routine indicated in the corresponding ADCON.	DMKDIR	READ SCANNAME SCAN1	
2	If the PROFILE control statement does not precede all USER control statements DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER name appears on the console preceded by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax.	DMKDIR	SCANPROF ERROR52	
	The SCANPROF routine creates a directory profile to include common control statements that can be referenced by each user's directory via the INCLUDE control statement.			
3	If the USER control statement follows a USER, ACCOUNT, OPTION, or IPL control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER user appears on the console followed by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax.	DMKDIR	 SCANUSER ERROR52	
	The last UDIRBLOK and UMACBLOK are masked off. Update the pointers to the buffers and write out the buffers that are full. The SCANUSER routine locates a UDIRBLOK and initializes it. Then the UMACBLOK is located and initialized.	DMKDIR		
4	If the INCLUDE control statement does not follow a USER control statement. DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER name appears on the console preceded by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax.		SCANINCL ERROR52	
	The SCANINCL routine updates the UMACBLOK by referencing the control statements contained in the directory profile.	DMKDIR		
5	If the CLASS control statement does not follow a USER control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER user appears on the console followed by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax.		SCANCLAS	
	The CLASSMAP routine creates a mask in UMACBLOK (UMACCLVL) to indicate the privilege classes allowed for this virtual machine.	DMKDIR	CLASSMAP	
6	If the ACCOUNT control card does not follow a USER, OPTION, or IPL control statement. DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER user appears on the console followed by the statement that was out of sequence.		SCANACCO ERROR52	
	Directory processing is terminated after scanning the remaining statements for syntax.			
	The SCANACCO routine updates the account number (UMACACCT) and distribution code (UMACDIST) fields of the UMACBLOK.			

Figure 17 (Part 1 of 2). DMKDIR Control Statement Processing

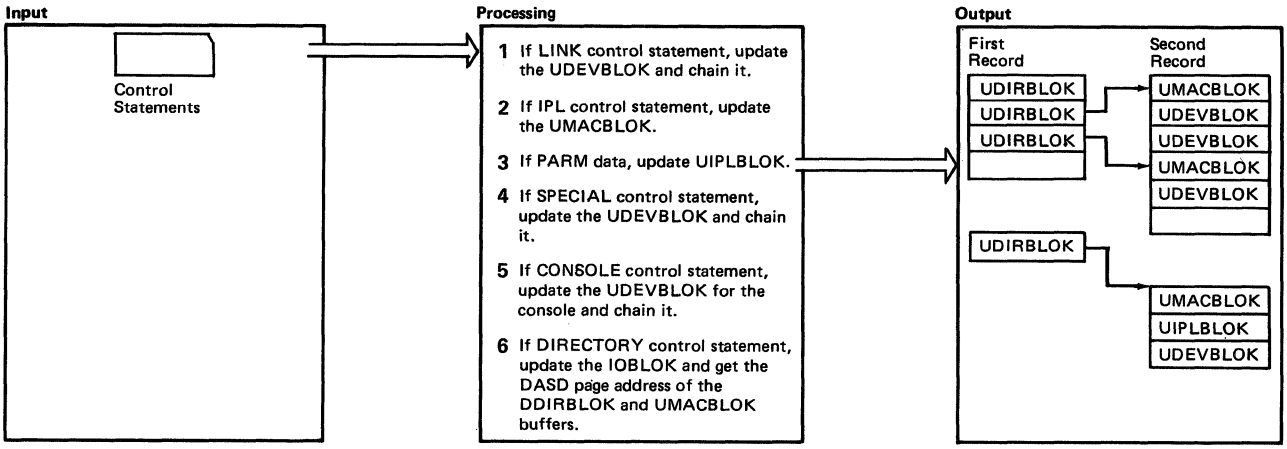
Notes	Module	Label	Ref
7 If the OPTION control statement does not follow a USER or IPL control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER user appears on the console followed by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax. The SCANOPTI routine sets fields in the UMACBLOK to indicate the machine options.		SCANOPT1 ERROR52	
8 The SCANMDIS routine branches and links to the SCANNAME routine with register 4 pointing to TABLE4. TABLE4 is scanned by device type to get the corresponding device class. The SCANMDIS routine then updates the device type (UDEVTYPE) and class (UDEVTYPC) fields in the UDEVBLOK. The UDEVSTATE field is updated to indicate a T-disk or long block, if either is present, and the number of cylinders is updated. For all disks other than T-disk, the volume serial number, mode, and password field of the UDEVBLOK are initialized. The mode is updated (except for a T-disl). **Label Value Comments** UDEVR 00 R link-mode UDEVRR 04 RR link-mode UDEVW 08 W link-mode UDEVWR 12 WR link-mode UDEVM 16 M link-mode UDEVMR 20 MR link-mode UDEVMW 24 MW link-mode The SCANMDIS routine that branches to the CHAINDEV routine to chain the UDEVBLOK to the UMACBLOK.		SCANMDIS SCANNAME SCANMDIS CHAINDEV	

Figure 17 (Part 2 of 2). DMKDIR Control Statement Processing



Notes		Module	Label	Ref
1	The SCANSPOO routine builds a UDEVBLOK. The UDEVSTAT field is set to X'08' to indicate a spool device. The virtual device address is stored in the UDEVADD field and the spool class is stored in the UDEVCLAS field. The SCANSPOO routine branches and links to the SCANNAME routine with register 4 pointing to TABLES. For all device types except the 2540, the spool class is picked up directly from TABLES. For a 2540 device, the device class is determined in the SCAN2540 routine. The default class is A, except for readers (readers default to class*).	DMKDIR	SCANSPOO	
	The SCANSPOO routine then branches to the CHAINDEV routine to chain the UDEVBLOK to the UMACBLOK.		CHAINDEV	
2	The SCANDEDI routine builds a UDEVBLOK. The UDEVSTAT field is set to X'80' to indicate a dedicated device. The virtual device address is stored in UDEVADD field. And, either the volume serial number (UDEVVSER) or user link to disk (UDEVLINK) fields are updated.	DMKDIR	SCANDEDI	
	The SCANDEDI routine then branches to the CHAINDEV routine to chain the UDEVBLOK to the UMACBLOK. The UDEVSTAT field is set to X'80' to indicate a dedicated device. The virtual device address is stored in UDEVADD field. And, either the volume serial number (UDEVVSER) or user link to disk (UDEVLINK) fields are updated.		CHAINDEV	
	The SCANDEDI routine then branches to the CHAINDEV routine to chain the UDEVBLOK to the UMACBLOK.			

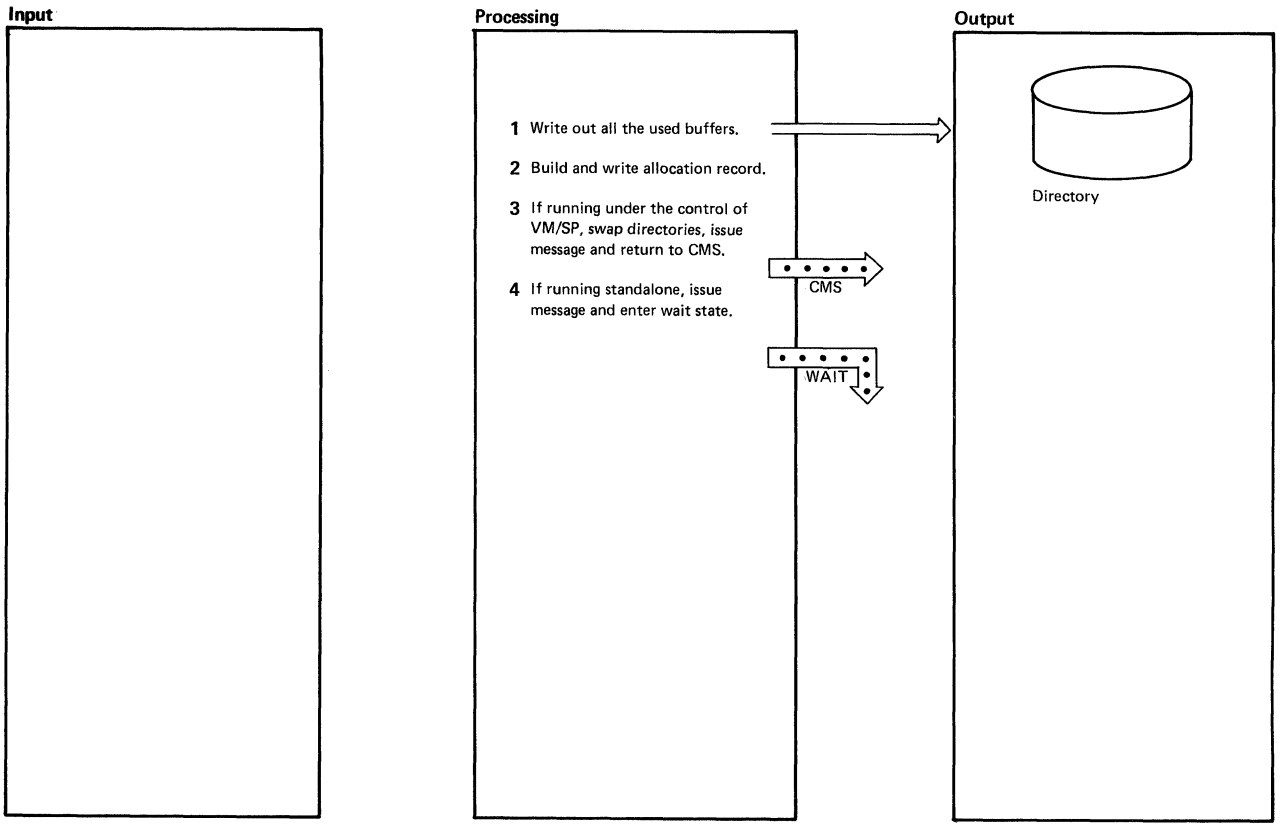
Figure 18. DMKDIR Control Statement Processing



Notes		Module	Label	Ref
1	The SCANLINK routine builds a UDEVBLOK. The UDEVSTAT field is set to X'10' to indicate the device is to be linked at logon time. The virtual device address (UDEVADD) and link device address (UDEVLINK) are updated. The mode (DEVMODE) is also updated. **Label / Value / Comments** UDEVR 00 R link-mode UDEVRR 04 RR link-mode UDEVW 08 W link-mode UDEVWR 12 WR link-mode UDEVM 16 M link-mode UDEVMR 20 MR link-mode UDEVMW 24 MW link-mode	DMKDIR	SCANLINK	
2	If the IPL control statement does not follow a USER, ACCOUNT, or OPTION control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING USER user appears on the console followed by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax. If there is no parm data field, the name of the system to be loaded (via IPL) at logon time is placed in the UMACIPL field of the UMACBLOK.	DMKDIR	SCANIPL ERROR52	
3	If there is a parm data field, fill in the UIPBLOK with the name of the system to be loaded, the length of the data, and the data. Update the UMACIPL field of the UMACBLOK to point to the UPIBLOK. Set code of X'00' in the UMACIPLX field to indicate that there is PARM data.			
4	The SCANSPEC routine builds a UDEVBLOK for a special device. The virtual device address is stored in UDEVADD, the device type is stored in UDEVTYPE, and the class is stored in UDEVTYPC. The SCANSPEC routine branches and links to the SCANNAME routine with register 4 pointing to TABLE2. The device type and class are picked up directly from TABLE2 for a 3270 and pseudo-timer. The SCANNAME routine branches (via an ADCON) to the SCANTCA. SCAN2701, SCAN2702, and SCAN2703 routines to determine the device type and class of channel-to-channel adapter, or 2701, 2702, and 2703 special device.	DMKDIR	SCANSPEC SCANNAME SCANCTA SCAN2701 SCAN2702 SCAN2703	
5	The SCANCONS routine builds a UDEVBLOK for the console. The virtual device address is stored in UDEVADD, the device type is stored in UDEVTYPE, and the class is stored in UDEVTYPC. The default class is T. The SCANCONS routine branches and links to the SCANNAME routine with register 4 pointing to TABLE3. The device type and class are picked up directly from TABLE3.	DMKDIR	SCANCONS	
6	If the DIRECTORY control statement is not the first control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING USER user The SCANDIRE routine sets up to update the IOBLOK. The output device address is stored in DASDADD, and the serial number is stored in DASDVSER. The DIRFLAG is set to a hexadecimal value that indicates the device type of the output unit. Then, the SCANDIRE routine gets the pointer to the first page of the directory and machine buffer areas.	DMKDIR ERROR52	SCANDIRE	

Figure 19. DMKDIR Control Statement Processing

Directory Program



Notes		Module	Label	Ref
1	All of the user directory, user machine, and user device buffers that were used are written. The buffers are written out by loading the DASD address into register 2, loading the buffer address into register 1, and then branching and linking to the WRITE routine.	DMKDIR	EXIT	
2	The allocate table is built. A table setting of X'OC' indicates an allocated cylinder. The VOL1 and allocation records are written.	DMKDIR	SCANALLO	
3	First the return PSW is set up and Registers 1 and 2 are set to the volume serial number. The user directories are swapped via a DIAGNOSE call to DMKUDRDS. The DIAGNOSE call to DMKUDRDS. The DIAGNOSE will program check if the user is not class A, B, or C. The directories are not swapped if the volume is not found in the OWNDLIST or if an I/O error occurs under CP. The message EOJ DIRECTORY UPDATED appears on the console and control returns to CMS. If no errors occur, and if the active system directory was updated, the directories are swapped. The message EOJ DIRECTORY UPDATED AND ON LINE appears on the console and control returns to CMS.	DMKDIR	MOVEPSW LOOP11	
4	If not running under VM/SP, the message EOJ DIRECTORY UPDATED appears on the console and the wait state is entered by loading the SVCNEW PSW.	DMKDIR	BARE	

Figure 20. Directory Exit

Program Organization

This section includes a program description of the DMKDIR module.

DMKDIR

Creates the VM/SP directory on a system owned volume.

Entry Points

DMKDIRCT is the entry point when the directory program is executed standalone.

DMKDIRED is the entry point when the directory program is executed under the control of CMS.

Routines Called

None

Attributes

Not serially reusable.

Registers at Exit

If executed under the control of CMS, register 15 contains a return code at exit.

Return Code	Meaning
1	Invalid filename or file not found
2	Error loading the directory
3	Invalid option from CMS
4	Directory not swapped, user class not A, B, or C
5	Directory not swapped, system (old) directory locked
6	Directory not swapped; the directory the system is using is not the directory just updated.
1xx	Error in CMS RDBUF routine. xx is the CMS routine return code.
2xx	Error in CMS TYPLIN routine. xx is the CMS routine return code.

Register Use

Reg	Use
R0	Work register
R1	Pointer to input field Pointer to IOB Pointer to output buffer Work register
R2	Input count from SCANCARD DASD address Work register
R3	Work register
R4	Work register
R5	Branch and link return address Pointer to the next UDEVBLOK Work register
R6	RDIRBUF, pointer to the UDIRBLOK buffer
R7	RMACBUF, pointer to the UMACBLOK buffer
R8	RDEVBUF, pointer to the UMDEVBLOK buffer
R9	Base register 3
R10	RMAC, pointer to UMACBLOK
R11	RDEV, pointer to UDEVBLOK
R12	Base register 1
R13	Base register 2
R14	Return address
R15	RDIR, pointer to UDIRBLOK.

External References

DMKURDS is called via a DIAGNOSE instruction to write the new VM/SP directory on DASD.

Directory

Following is an alphabetic list of the major labels of the Directory program. The associated method of operation diagram is referenced and a brief description of the function performed at the point in the program corresponding to each label is included.

Label	Figure	Description
BARE	20	Directory program exit when not running under the control of VM/SP.
BILDUDIR		Builds UDIRBLOK.
BILDUMAC		Builds UMACBLOK.
BINCONV		Converts decimal numbers to binary.
CHAINDEV	17, 18	Chains UDEVBLOK to UMACBLOK.
CLASSMAP	17	Builds a mask in UMACBLOK indicating privilege classes allowed for this virtual machine.
CMS1	16	Sets up the parameter list identifying the file containing the control statements when running under CMS.
CMS3		Reads CMS control cards via SVC 202.
COMMERRP		Prints queued error messages.
COMPARE		Compares keywords and sets condition codes.
DECCONV		Converts decimal numbers to hexadecimal.
DEFAUL13	16	Defaults to the IPL device for control statement input device when running standalone.
DMKDIRCT	16	Sets up base registers and initializes pointers.
EDITTEST	16	Sets DIRFLAG to X'20' to indicate edit, if EDIT is specified when the Directory program is run under VM/SP.
EOF		Simulates a USER card.
ERROR51		Error processing for invalid operand.
ERROR52	17, 19	Issues message when a control statement is out of sequence.
ERROR58		Issues message DMKDIR758E.
ERROR62		Issues message DMKDIR762E.
EXIT	16, 20	End-of-job processing for Directory Program.
GETALT		Makes switch from first to second device address specified.
GETCYLNO		Fills in cylinder relocation for minidisks.
GETPAGE		Assigns a DASD page address.
GRAPHID	16	Reads the input control statements from a display terminal when the directory program is not running under CMS.
HEXCONV		Converts hexadecimal numbers to binary.
LONG		Turns on long block indicator for minidisks.
LOOP11	20	Calls DMKUDRDS via the DIAGNOSE instruction to swap directories when running under VM/SP.
MOVECPT		Sets up current control statement pointer.
MOVEDISP		Updates UMACBLOK.
MOVEPSW	20	Sets up return PSW before issuing DIAGNOSE to call DMKUDRDS.
MSGRET	16	When running standalone, a header line is printed.
MSG02A	16	Requests input device when running standalone.
MSGWRITE		Writes messages to the terminal.
NOTUSED		Updates UMACBLOK pointer.
POINTDEV		Updates UDEVBLOK pointer.
READ	16, 17	Reads control statements and branches to appropriate processing routine.

Label	Figure	Description
REREAD		Sets up pointer to control statement read buffer.
RET1		Scans control statements.
SCANACCO	17	ACCOUNT statement processing routine.
SCANALLO	20	Builds allocation record.
SCANCARD		Scans the control statement for the next operand.
SCANCLAS	17	Processes the CLASS control statement.
SCANCONS	19	CONSOLE statement processing routine.
SCANCTCA	19	Updates the UDEVBLOK and chains the control unit to the UDEVBLOK for channel-to-channel adapters.
SCANDEDI	18	DEDICATE statement processing routine.
SCANDIRE	19	DIRECTORY statement processing routine.
SCANPROF	18	PROFILE statement processing routine.
SCANIPL	19	IPL statement processing routine.
SCANLINK	19	LINK statement processing routine.
SCANMDIS	17	MDISK statement processing routine.
SCANNAME	17, 19	Scans the name table until a match is found. Register 4 points to the name table. If the name field is a constant, it is put in the UDEVBLOK. If the name field is an address, control is passed to that address.
SCANOPTI	17	OPTION statement processing routine.
SCANINCL	18	INCLUDE statement processing routine.
SCANSPEC	19	SPECIAL statement processing routine.
SCANSPOO	18	SPOOL statement processing routine.
SCANUSER	17	USER statement processing routine.
SCAN1	17	Points register 4 to TABLE1, then branches and links to SCANNAME processing routine.
SCAN2311		Updates the UDEVBLOK for 2311 disks.
SCAN2540		Updates the UDEVBLOK for 2540 devices.
SCAN2701	19	Updates the UDEVBLOK for 2701 devices.
SCAN2702	19	Updates the UDEVBLOK for 2702 devices.
SCAN2703	19	Updates the UDEVBLOK for 2703 devices.
STARTIO	16	Reads the input control statements if the directory program is not running under CMS.
STATE	16	Checks that control statement file exists.
STOREADD	16	Sets the DIRFLAG to X'20' to indicate edit, if EDIT is specified when the Directory program is run standalone.
TERM	16	At end of processing, returns control to CMS if running under VM/SP.
TESTBUFF		Tests to see if UDEVBLOK was used.
TESTUDEV		Gets DASD address of UMACBLOK.
UPDATE		Points to next UDEVBLOK.
UPDATECT		Updates device count in UMACBLOK.
WRITE		Writes the directory on DASD.
XERR754E		Keeps track if first address on DIRECT statement not operational.
XERR755E		Keeps track of I/O errors.
XERR761E		Keeps track of volid.

Data Areas

The directory exists on disk as 4K (page size) records. The VOL1 label (cylinder 0 track 0 record 3), on the volume containing the directory, points to the directory. The directory starts with the first available record.

The first UDIRBLOK is a dummy UDIRBLOK. Its UDIRDISP field points to the last UDIRBLOK in that record. The UDIRDASD field points to the next UDIR record, or, if it is the last record, it contains zeros. The second UDIRBLOK in the first record points to the UMACBLOK for that user, located in the second record. In turn, the UMACBLOK points to the first UDEVBLOK for that user. It is the second block in the second record. The last UDEVBLOK for this user has a pointer of all zeros.

The directory entry for the second user consists of a UDIRBLOK in the first record and associated UMACBLOK, and UDEVBLOKs in the second record. When a record becomes full, the chain continues into the next available record.

When the directory is created, all UDIRBLOKs are grouped 169 blocks per record. The UMACBLOK and UDEVBLOKs are sequentially chained into a separate record. If the record becomes full before the end of the chain, the chain overflows into the next available record.

The formula to find the number of records is:

$$\frac{NU}{169} + \frac{((NU+NM)*2)+ND}{170} = NR$$

where:

NU is the number of user records.
NM is the number of MDISK cards describing a virtual disk (not T-Disk).
ND is the total number of MDISK (describing T-Disk space), SPOOL, LINK, SPECIAL, CONSOLE, and DEDICATE cards.
NR is the total number of records used.

For count-key-data DASD, to find the number of cylinders, divide the total number of records by 32 for 2314/2319 devices, by 57 for 3330 series devices, by 24 for 3340 and 2305 series devices, by 96 for the 3375, or by 150 for the 3380. If you install the speed matching feature (Feature #6550) with the 3380, the extended count-key-data channel programs will be used. For FB-512 DASD, the total number of pages needed equals NR. To ensure that a new directory will not overlap an existing directory, allow space for two directories or allocate a new directory each time the directory is created.

Directory Program

The following data areas are used by the directory program:

- The UDEVBLOK (user device block), built in the UDEVBLOK or UMACBLOK buffer.
- The UDIRBLOK (user directory block), built in the DIRBLOK buffer.
- The UMACBLOK (user machine block), built in the UMACBLOK buffer.

These data areas, as well as a figure showing the user directory format and the relationship of the above blocks, are described in the *VM/SP Data Areas and Control Block Logic Volume 1 (CP).*

Diagnostic Aids

Following is a list of the messages issued by the Directory program. The label of the message and the associated method of operation diagram are included in the list.

Message Code	Label	Figure	Message Text
DMKDIR536I	ERR536		rdev REPORTS DISABLED INTERFACE; FAULT CODE = code; NOTIFY CE
DMKDIR750E	ERROR50		RESTRICTED PASSWORD FILE, ERROR DURING READ
DMKDIR750E	ERROR50		RESTRICTED PASSWORD FILE HAS BAD RECORD FORMAT
DMKDIR750E	ERROR50		RESTRICTED PASSWORD FILE HAS BAD RECORD LENGTH
DMKDIR750W	ERROR50		RESTRICTED PASSWORD FILE NOT FOUND
DMKDIR751E	ERROR51A		INVALID OPERAND - operand
DMKDIR752E	ERROR52	17, 19	STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER user
DMKDIR753E	ERROR53		OPERAND MISSING
DMKDIR754E	ERROR54A STARTIO READ WRITE		DEVICE rdev NOT OPERATIONAL
DMKDIR755E	ERROR55A WRITE		I/O ERROR rdev CSW=csw SENSE=sense
DMKDIR756E	ERROR56A		PROGRAM CHECK PSW = psw
DMKDIR757E	ERROR57		MACHINE CHECK RUN SEREP AND SAVE OUTPUT FOR CE
DMKDIR758E	ERROR58 CHAINDEV		DUPLICATE UNIT DEFINITION
DMKDIR760E	ERROR60 GETPAGE		NOT ENOUGH SPACE ALLOCATED FOR DIRECTORY
DMKDIR761E	ERROR61A SCANDIRE		VOLID READ IS volid1 NOT volid2 ON rdev
DMKDIR762E	ERROR62 READ		DIRECTORY STATEMENT MISSING
DMKDIR763E	ERROR63 STATE	16	DIRECTORY FILENAME INVALID OR FILE NOT FOUND
DMKDIR764E	ERROR64		ERROR IN routine
DMKDIR765E			INVALID CLASS DEFINITION
DMKDIR766E			DUPLICATE CLASS DEFINITION
DMKDIR767W	ERROR67		PASSWORD CHANGED TO NOLOG FOR USERID
DMKDIR768E			FOR userid - MOVE vdev TO A (NON)SHARED VCU
	MSG04	16	EOJ DIRECTORY NOT UPDATED
	MSG01	20	EOJ DIRECTORY UPDATED
	MSG03	20	EOJ DIRECTORY UPDATED AND ON LINE

Message Code	Label	Figure	Message Text
	MSG02	16	VM/370 USER DIRECTORY CREATION PROGRAM
	MSG02A	16	ENTER CARD READER DEVICE ADDRESS AND OPTIONS
DMKDIR771E	ERROR71		RESTRICTED PASSWORD AND NOLOG INVALID FOR userid
DMKDIR782E	FREERR		ERROR ATTEMPTING TO GET FREE STORAGE
DMKDIR782I	EXIT		ERROR ATTEMPTING TO RETURN FREE STORAGE

Directory Program

Index

Directory Program

Chapter 3. DASD Dump Restore Program

Introduction

The DASD Dump Restore program can execute stand-alone or under the control of CMS via the DDR command. It performs five functions for Direct Access Storage Devices. The five functions are:

- Dump
- Restore
- Copy
- Print
- Type.

These functions apply to both count-key-data DASD and to FB-512 DASD.

DDR can store data on tape in a compact format. DDR does this by compressing strings of duplicate data into a smaller amount of space and reducing the amount of space necessary to represent the characters in the data. This uses less tape space than the standard format. The compact format is an option (COMPACT) specified by the user on the OUTPUT control statement for the dump function. If it is used on the OUTPUT control statement for the copy function, a system message is issued saying the COMPACT option is ignored. It is valid only for tape output. Tapes created by DDR which are in the compact format may be used as input to the restore, copy, print, and type functions.

Note: Tapes created by DDR which are in the compact format cannot be used as input to earlier levels of DDR.

The 8809 tape device has a characteristic called the streaming mode. This device has two programmable tape speeds or modes. In start-stop mode (normal speed) the tape travels at 12.5 inches per second and stops in the interrecord gap. In streaming mode (high speed) the tape travels 100 inches per second and does not stop in the interrecord gap. It traverses the gap at high speed in anticipation of the next SIO. If this SIO does not occur in time, the tape goes past the start of the record.

The technique to provide streaming is called double buffering with I/O overlap. That means two I/O buffers are used concurrently and DDR does not wait for I/O to complete before resuming processing.

Note: The 8809 tape drive may not operate efficiently in streaming mode while DDR is processing data in the compact format.

DUMP

The dump function saves data from a direct access volume on magnetic tape. The output tape may be put into compact format. For the count-key-data (non-full track read and non-compact format), the data is saved cylinder by cylinder. The format of the tape is:

- Record 1, volume header record — data describing the volume.
- Record 2, track header record — a list of count fields to restore the track and the number of data records written on tape. After the last count field the record contains key and data records to fill the 4K buffer.
- Record 3, track data records — key and data records packed into 4K blocks with the last block truncated.
- Record 4, either the end-of-volume or end-of-job trailer label. The end-of-volume label contains the same information as the next volume header record except that the ID field contains EOV. The end-of-job trailer label contains the same information as record 1 except that the cylinder number field contains the disk address of the last record on tape and the ID field contains EOJ.

FB-512

For FB-512 devices (compact or non-compact format) the data is saved in groups of FB-512 blocks. Any number of blocks can be dumped. The format of the tape is:

- Record 1, volume header record — data describing the volume.
- Record 2, track header record — data describing the group of FB-512 data blocks that follow, as well as the number of tape records required to hold these blocks. After the control data, the record contains FB-512 data to fill the 4K buffer.
- Record 3, FB-512 data records — contains the FB-512 blocks dumped from the FB-512 volume.
- Record 4, either the end-of-volume or end-of-job trailer label. The end-of-volume label contains the same information as the next volume header record except that the ID field contains EOV. The end-of-job trailer is the same as the EOV label except that the ID contains EOJ and the block-number field contains the number of the last block on the tape.

For count-key-data (full track read format or compact format) the data is saved cylinder by cylinder as follows:

- Record 2, track header record — length of track, density of the tape, and number of count fields in the track followed by track contents.

- Record 3, track data records — count-key-data records in 8K, 12K, or 49K blocks for 800 bpi, 1600 bpi, or 6250 bpi tapes respectively, or 49K blocks for the 3480 Tape Subsystem. The last block is a short block.

- Record 4, either the end-of-volume or end-of-job trailer label. The end-of-volume label contains the same information as the next volume header record except that the ID field contains EOV. The end-of-job trailer label contains the same information as record 1 except that the cylinder number field contains the disk address of the last record on tape and designates the allocation of cylinders and the ID field contains EOJ.

RESTORE

The restore function transfers data from a tape created by the DDR dump function to a DASD device. The data may be restored only to a device of the same type as the device from which it was dumped. A tape in compact format may be used as input. DDR checks to see if the input is in compact format, and expands the data, if needed.

COPY

The copy function copies data from one device to another device of the same type (DDR does not copy from count-key-data to FB-512 or from FB-512 to count-key-data). For disk-to-disk operations, data may be reordered on a cylinder or block basis. If copying from tape-to-tape, the input tape must have been created by the DDR dump function.

A tape in compact format may be used as input. For a tape-to-tape copy, the output tape will be in the same format (compact or standard) as the input tape. The COMPACT option on the OUTPUT control statement is not valid for the COPY function. If it is specified, a message stating, 'COMPACT OPTION IGNORED FOR COPY OPERATIONS' is displayed.

PRINT

The print function prints both hexadecimal and EBCDIC representations of selected records of a DASD or Tape Volume on a printer. The word "record" here means a particular block when referring to FB-512 data. A tape in compact format may be used as input.

TYPE

The type function displays at the terminal both hexadecimal and EBCDIC representations of selected records or blocks of a DASD or tape volume. A tape in compact format may be used as input.

Method of Operation

The method of operation diagrams describe the major functions of the DDR (DASD Dump Restore) program. The relationship of the method of operation figures is described in Figure 21.

The five functions of DDR apply equally to FB-512 data as they do to count-key-data devices. The method of operation for each is the same at a given level of description. The main difference is the unit of DASD data that DDR can handle for FB-512 devices. This unit is an FB-512 block. This means that DDR can copy, dump, restore, print, or type any number of blocks in increments as small as one block. For count-key-data, the unit of copy, dump, or restore is one cylinder; for print or type it is one record. This difference leads to a different control statement format, as well as different internal processing. This distinction is noted in the following diagrams, as appropriate.

Figure 22 describes the major functions of the DDR program.

Figure 23 shows the control statement processing for the DDR program.

Figure 24 describes the Dump function.

Figure 25 describes the Dump function with streaming.

Figure 26 describes the Restore function.

Figure 27 describes the Restore function with streaming.

Figure 28 describes the Copy function.

Figure 29 describes the Print function.

Figure 30 describes the Type function.

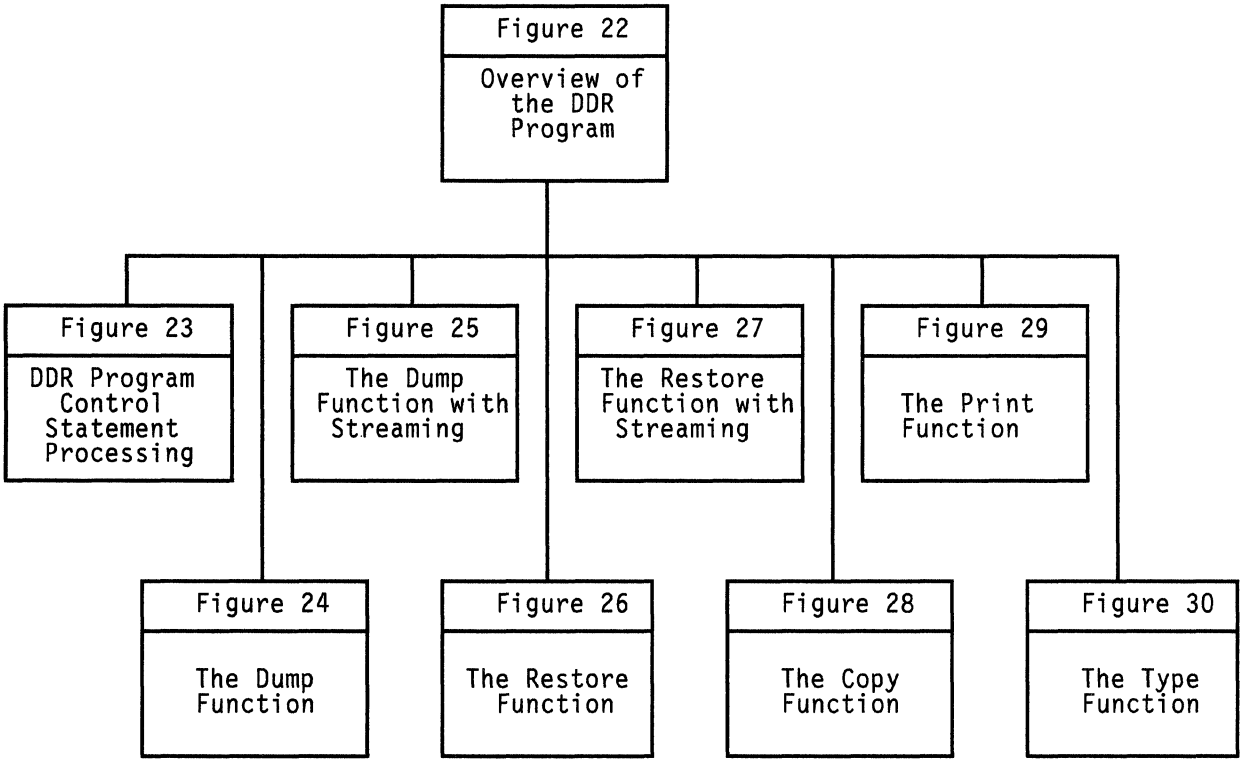
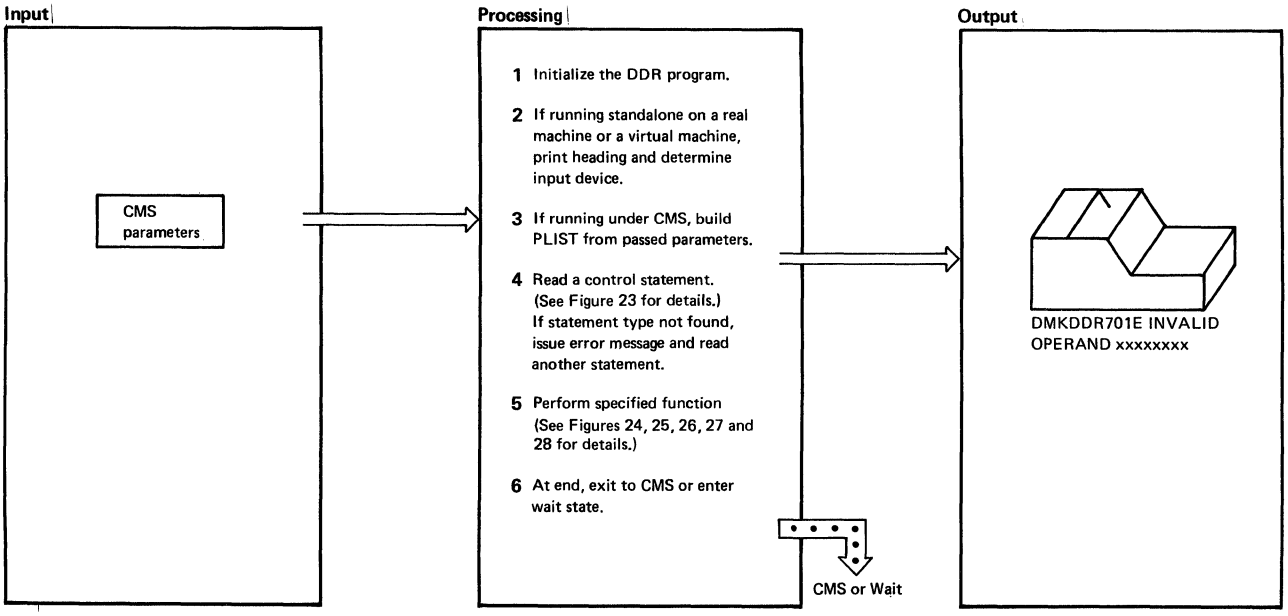


Figure 21. Key to the DASD Dump Restore Program Method of Operation Diagrams

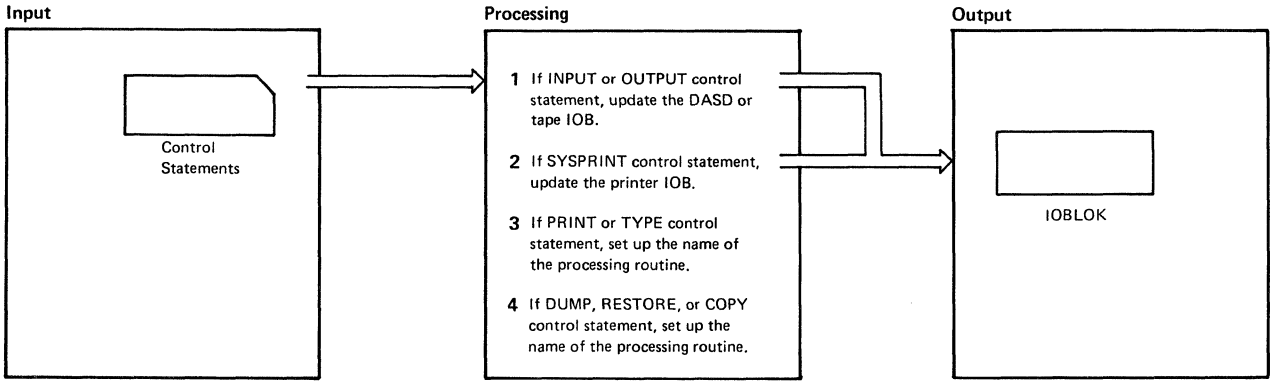
DASD Dump Restore Program



Notes		Module	Label	Ref
1	The DDR program is initialized and the base registers (9, 10, 11, 12, and 13) are set up. Register 8 is initialized to the data buffer address.	DMKDDR	DMKDDREP	
2	The heading VM/370 DASD DUMP/RESTORE PROGRAM RELEASE n is displayed. If no input device is specified, the IPL device is used as the input device.	DMKDDR	NEWADD	
3	DMKDDR builds a PLIST if parameters are passed from CMS to the DDR program.	DMKDDR	CMS1	
4	DMKDDR reads the control statement. The routine needed to initialize the DDR function is found by branching and linking to the SCANNAME routine and searching the name table.	DMKDDR	GTCARD	
5	The designated function is performed. At its end, control returns to the GTCARD routine to read the next control statement and perform the next function.	DMKDDR		
6	When the last control statement is read and processed, the GTCARD routine branches to the EXIT routine.	DMKDDR	EXIT	
	The end of job statement (MSG001 is displayed. If running under CP, the SYSPRINT device is closed and control returns to the CMS command environment.		CMS8	
	If running standalone, the wait state is entered.		TESTCMS	

Figure 22. Overview of the DDR Program

DASD Dump Restore Program



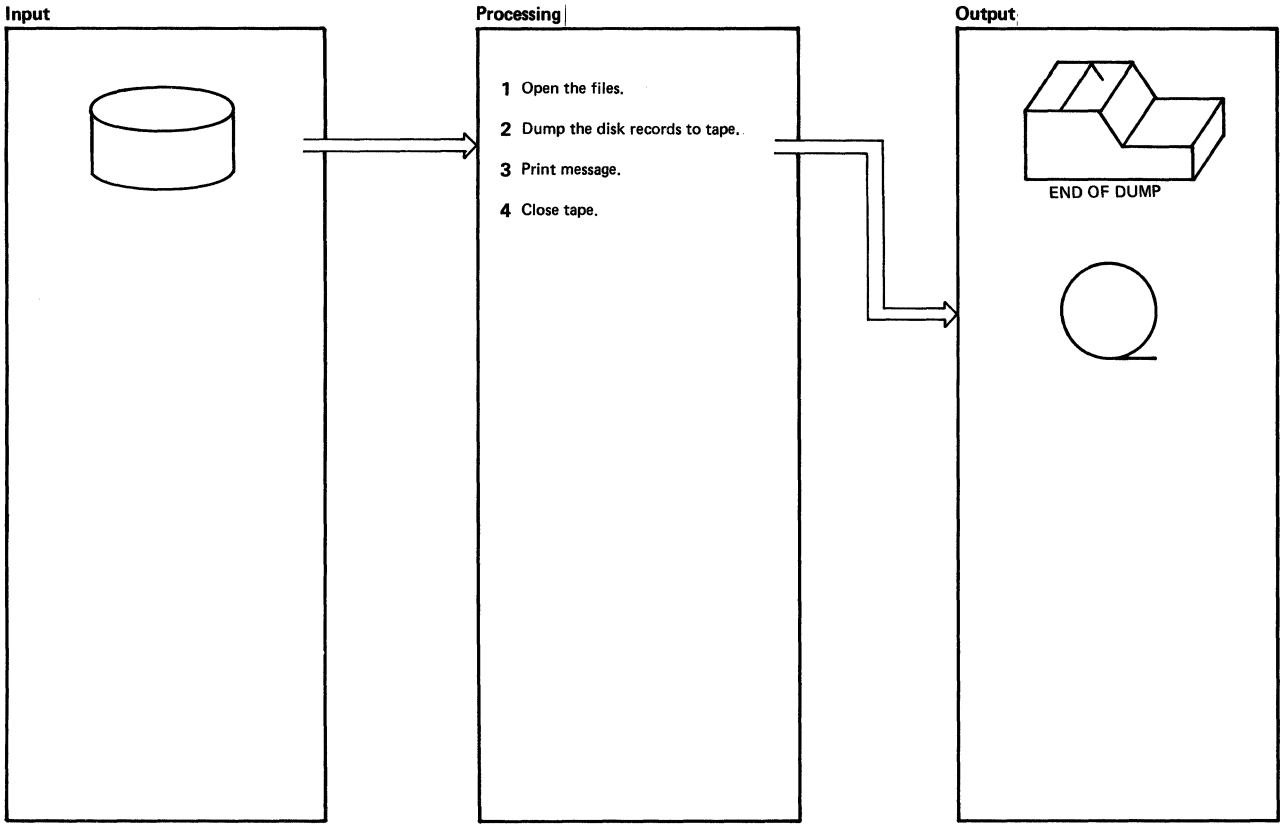
Notes	Module	Label	Ref
1 The address of the IOB is loaded into register 15. DMKDDR gets the unit address. The unit address (IOBUADD) and alternate tape address (IOBATAPE) fields of the IOB are filled in. DMKDDR reads the device type from the control statement and then branches and links to the SCAN routine. The SCAN routine searches a table of valid devices and picks up the device class and type. The class (IOBCLASS) and type (IOBTYPE) fields are updated. The codes for the various device classes are contained in the DEVTYPES COPY file. If a DASD serial number is specified, the volume serial number (IOBVSER) field is updated. If tape options are specified, the IOB is updated. **Field** **Options** IOBSKIP number of times file to be forward spaced. IOBMODE tape mode. X'C3' indicates 9 track 1600 BPI X'CB' indicates 9 track 800 BPI X'DB' indicates 18 track 38K BPI X'D3' indicates 9 track 6250 BPI IOBDISP disposition of tape. X'07' indicates rewind X'OF' indicates rewind and unload X'03' indicates tape is not to be repositioned Either of the following error messages may be displayed while processing INPUT or OUTPUT control statements. DMKDDR701E INVALID OPERAND - operand DMKDDR703E OPERAND MISSING If either of these errors occurs, the control statement is ignored and control returns to the CTCARD routine to read the next control statement.	DMKDDR	SCANINPU SCANOUTP	
2 The address of the printer IOB is loaded into register 15. The printer unit address is placed in the IOBUADD file of the IOB. If an error occurs, either message DMKDDR701E INVALID OPERAND - operand DMKDDR703E OPERAND MISSING is displayed. The statement in error is ignored, and control returns to the CTCARD routine to read the next control statement.	DMKDDR	SCANSYP	
3 The translate table is set up. If TYPE is specified, the LOWERCAS table is used. If PRINT is specified, the UPPERCAS table is used. The routine name is set up: PRINT or TYPE. The start address (default is track 0 record 0 or block 0 for FB-512), and the stop address (default is last track and last record or the last block for FB-512) are set up. If TYPE is specified, the console skips one line. In PRINT is specified, the printer skips to channel 1. If there is an error in the control statement, either error message DMKDDR701E INVALID OPERAND - operand DMKDDR703E OPERAND MISSING is displayed. The control statement is ignored, and the next control card is read by the GTCARD routine.	DMKDDR	SCANPRIN SCANTYPE	

Figure 23 (Part 1 of 2). DDR Program Control Statement Processing

Notes		Module	Label	Ref
4	If DUMP control statement, set the processing routine name to DUMP.	DMKDDR	SCANDUMP	
	If RESTORE control statement, set the processing routine name to RESTORE.		SCANREST	
	If COPY control statement, set the processing routine name to COPY.		SCANCOPY	
	For the dump function, the input must be a DASD and output a tape.			
	For the restore function, the input must be a tape, the output a DASD.			
	For a copy function, the input and output devices must be the same class and type. If the input device contains more cylinders of blocks than the output device, the following message is issued: DMKDDR725R DASD INPUT DEVICE WAS (IS) LARGER THAN OUTPUT DEVICE. DO YOU WISH TO CONTINUE? RESPOND YES OR NO:			
	The operator must determine if the copy function is to continue.			
	COMPACT OPTION		DDR731	
	For the dump function, the COMPACT option is valid on the OUTPUT control statement for a tape device.			
	The COMPACT option is ignored if it is specified on the INPUT control statement for a tape device.			
	For the copy function (tape-to-tape) if the COMPACT option is specified on the OUTPUT control statement, the following message is issued and processing of the copy function continues: DMKDDR731I COMPACT OPTION IGNORED FOR COPY OPERATIONS			
	The output tape is produced in the same format (compact or noncompact) as the input tape.			

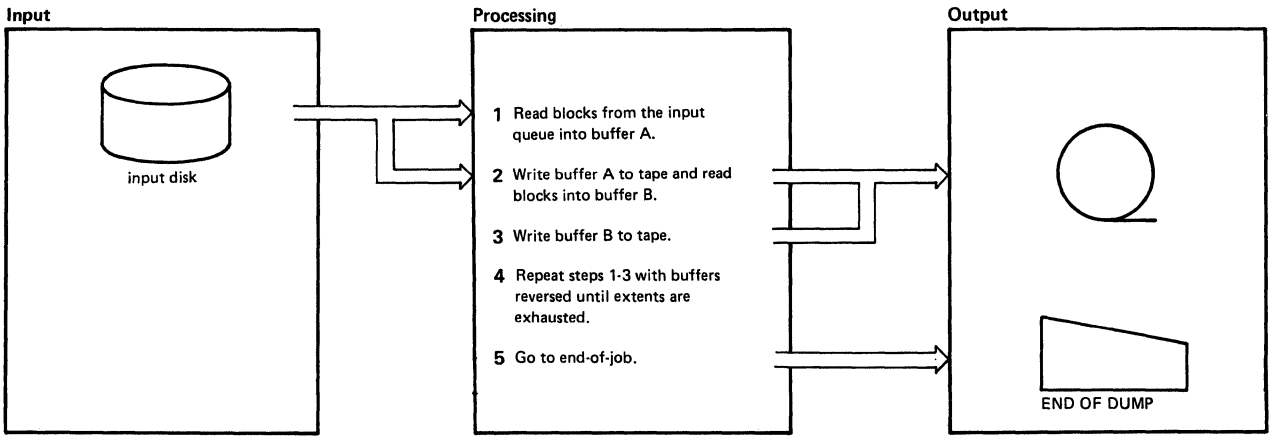
Figure 23 (Part 2 of 2). DDR Program Control Statement Processing

DASD Dump Restore Program



Notes		Module	Label	Ref
1	The input disk is opened by branching and linking to the OPENDASD routine. The extent table is updated to define the extents to be dumped. Each statement updates the extent table until a null line, an INPUT statement, or OUTPUT statement is read. top The output tape is opened, the proper number (if any) of records is skipped and the volume header record (VHR) is written.	DMKDDR	OPENIN GETEXT OPENOUT	
2	Prints the headings indicating the function being performed and the date and time of the dump. The read, write, and update cycle continues until the indicated disk extents are dumped to tape. Starting at the first disk extent (CYLSTART or BLKSTART), the disk records are read. The record is written on tape and the pointers are updated to the next disk record. If the COMPACT option is specified, a branch is made to the encoding routines and data is written in compact format. The dump cycle continues until the last disk extent CYLSTOP or BLKSTOP, is dumped to tape.	DMKDNC DMKDNT DMKDDR	CCMP300 TESTOUT UPDTADD	
3	The message END DUMP indicates that the dump function has successfully terminated. If the COMPACT option was specified, the following messages are displayed: BYTES IN__BYTES OUT__ TRACKS NOT COMPACTED ON THE TAPE.__ BLOCKS NOT COMPACTED ON THE TAPE.__		CCMP200	
4	The trailer record is written on the output tape. If the tape disposition was specified on the DUMP control statement, the tape is so positioned now. Control returns to the control statement read routine (GRCARD) to read and process the next control statement.	DMKDDR	EOJ	

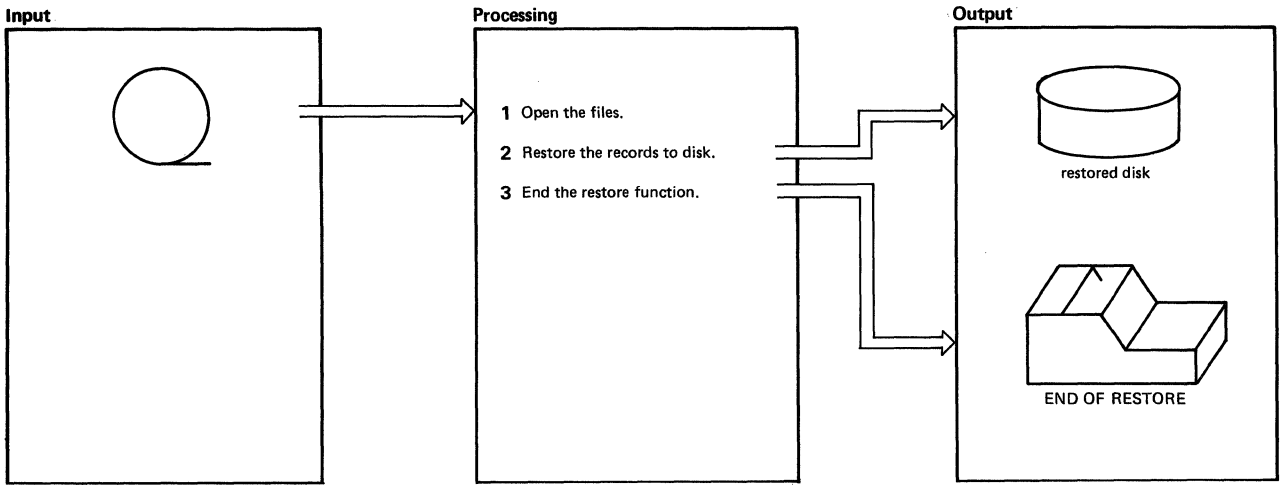
Figure 24. Dump Function



Notes		Module	Label	Ref
1	For the dump function, the work queues are initialized so that the DASD input queue (Q2) owns both buffers. Therefore, Q2 is selected first. A channel program is built to read blocks from the input queue. The number it reads is the number that will fit in the buffer or the number that remains in the current extent of the input disk (whichever is smaller). It starts reading from the block number in INBLADD. It sets control fields in the THR to describe the data as follows: THRFRSBL - the block number of the first block read THRLASBL - the block number of the last block read THRNBLK - the number of blocks to be written THROBLAD - the number to be assigned to the first block when it is output	DMKDDR DMKDDR	QSEARCH FBAIN	
	FBAIN calls the start I/O routine for overlapped I/O. The SIO is issued. When cc=0, control is passed to the caller (FBAIN). FBAIN passes control to QSEARCH. QSEARCH waits because there is no work; the output tape has no work and the input DASD is busy. When the FB-512 device interrupts, the I/O interruption routine goes to the interruption return address (FINIRA). Control fields in the DDR are updated in preparation for the next input operation. The fields are: INBLADD - the block number of the next block to be read from tape OUTBLADD - the block number to which INBLADD should be written.	DMKDDR DMKDDR	QSEARCH FINIRA	
2	If the blocks just read completed an extent, another field (CUREXT) is updated to point to the next entry in the extent table. If there are no more blocks, return is to ENQBUF. This routes the now filled buffer to the output queue, but does not return INIOB to the input queue. This temporarily precludes further input.			
	The IOB address (R15) is stored in the appropriate queue. The INIOB address is stored in the DASD or input queue(Q1). Then the THR address (IOBTHR) is stored in the appropriate buffer list. The address of buffer A is stored in the output's list (B2) to signal that a buffer is ready to be output. Both IOBs are available, and a buffer is also ready on both queues. Because the tape queue is inspected first, FBAIN gets control to read into buffer B.	DMKDDR DMKDDR DMKDDR	ENQIOB QSEARCH FBAIN	
	A CCW string is built to write the THR. The write CCWs are for 4K each (the last one is short) and are command chained together. STARTIOO (the overlapped entry) is called to do the I/O. When the condition code from the SIO is 0, control is returned to the caller (TAPOUT). Then control is immediately passed to QSEARCH. QSEARCH selected FBAIN. The SIO is issued and, when cc=0, QSEARCH is reentered. QSEARCH waits because there is no work to do.	DMKDDR DMKDDR DMKDDR DMKDDR DMKDDR DMKDDR	TAPOUT STARTIOO QSEARCH FBAIN QSEARCH FINIRA	
3	The FB-512 device interrupts. The I/O interruption routine goes to FINIRA. Control fields are updated as before and the IOB and the buffers are enqueued as before. QSEARCH waits for work. The tape interrupts and the I/O interruption routine goes to IOBIRA (CHKEOE).	DMKDDR DMKDDR DMKDDR	ENQIOB QSEARCH	
4	CHKEOE determines if the THR just output is the last one to be output in a given extent. If not, return is to ENQIOB. If it is, a call is made to PRINTEXT to print the extent map. If this is the last THR extent, control is passed to the EOJ routine. Otherwise, control passes to ENQIOB. The tape IOB is enqueued and the buffer is routed to the output queue. Both input and output routines are eligible. Because the tape queue is inspected first, TAPOUT is given control. Steps 1 through 3 are repeated.	DMKDDR DMKDDR DMKDDR	CHKEOE ENQIOB QSEARCH	
5	The cycle continues until the extents are exhausted. Then the input is turned off. At the completion of the next output, control is passed to the EOJ routine.			

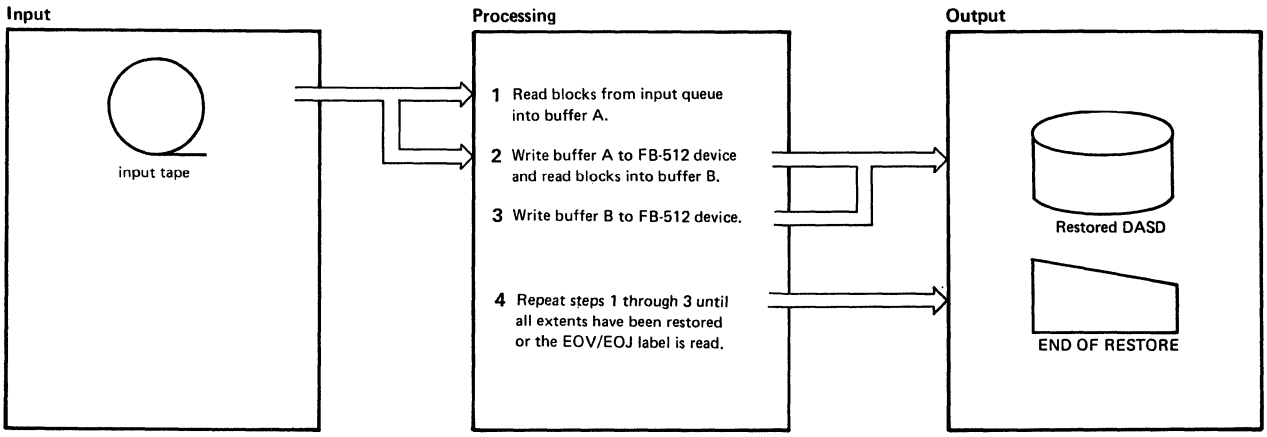
Figure 25. Dump Function with Streaming

DASD Dump Restore Program



Notes		Module	Label	Ref
1	The input tape is opened and positioned if the RESTORE control statement specified that records were to be skipped.	DMKDDR	OPENIN	
	A check is made to ensure that the output disk has the correct volume serial number. If the volume serial number is incorrect, the message DMKDDR717R DATA DUMPED FROM volid1 TO BE RESTORED TO volid2. DO YOU WISH TO CONTINUE? RESPOND YES, NO OR REREAD: is displayed. The operator must decide if the restore function is to continue.		SETDASD	
	The extent table is updated to indicate the extents to be restored to disk. The output disk is opened by branching and linking to the OPENDASD routine.		GETEXT OPENOUT	
2	The headings are printed, indicating that the restore function is starting.	DMKDDR	PRINTD MSG004	
	The number of cylinders or blocks on the original DASD input device is compared with the number of cylinders or blocks on the DASD output device. If the input device was larger, the following message is issued: DMKDDR725R DASD INPUT DEVICE WAS (IS) LARGER THAN OUTPUT DEVICE. DO YOU WISH TO CONTINUE? RESPOND YES OR NO: The operator must determine if the restore function is to continue.			
	The read and write loop continues until all the specified extents are restored to disk. The tape records are read from the tape that has been positioned.		GETTHR	
	A check is made to see if the tape is in compact format. If it is, the date is decoded, if there is an error during decoding, the message DMKDDR728E DECODE ERROR ENCOUNTERED: nn is displayed. 'xx' is the return code from the decoding routine. It can have the following values: 2 First byte of input is 0 or is greater than 5. This should not occur. It may be caused by using a set of encoding tables which do not match the decoding tables which are supplied. 3 There is more data to be decoded, but the output buffer is not big enough to hold more. Decoding stopped when the output buffer became full. 4 The decoding tables are malformed or the data in compact format was incorrectly transmitted. The program tried to decode a code word which could not be decoded within its first 21 bits.	DMKDDC DMKDDT	DCMP500	
	The data is written on the indicated disk cylinders or blocks and the pointers to the disk are updated for the next record. The restore function is complete when the last cylinder (CYCLSTOP) or block (BLKSTOP) is restored.	DMKDDR	DASDWRIT UPDTADD	
3	The message END OF RESTORE is displayed. If the data was decoded from compact format, the following message is displayed: BYTES RESTORED Control returns to the GTCARD routine to read the next control statement.	DMKDDR	CLOSEJOB CCMP200	

Figure 26. Restore Function



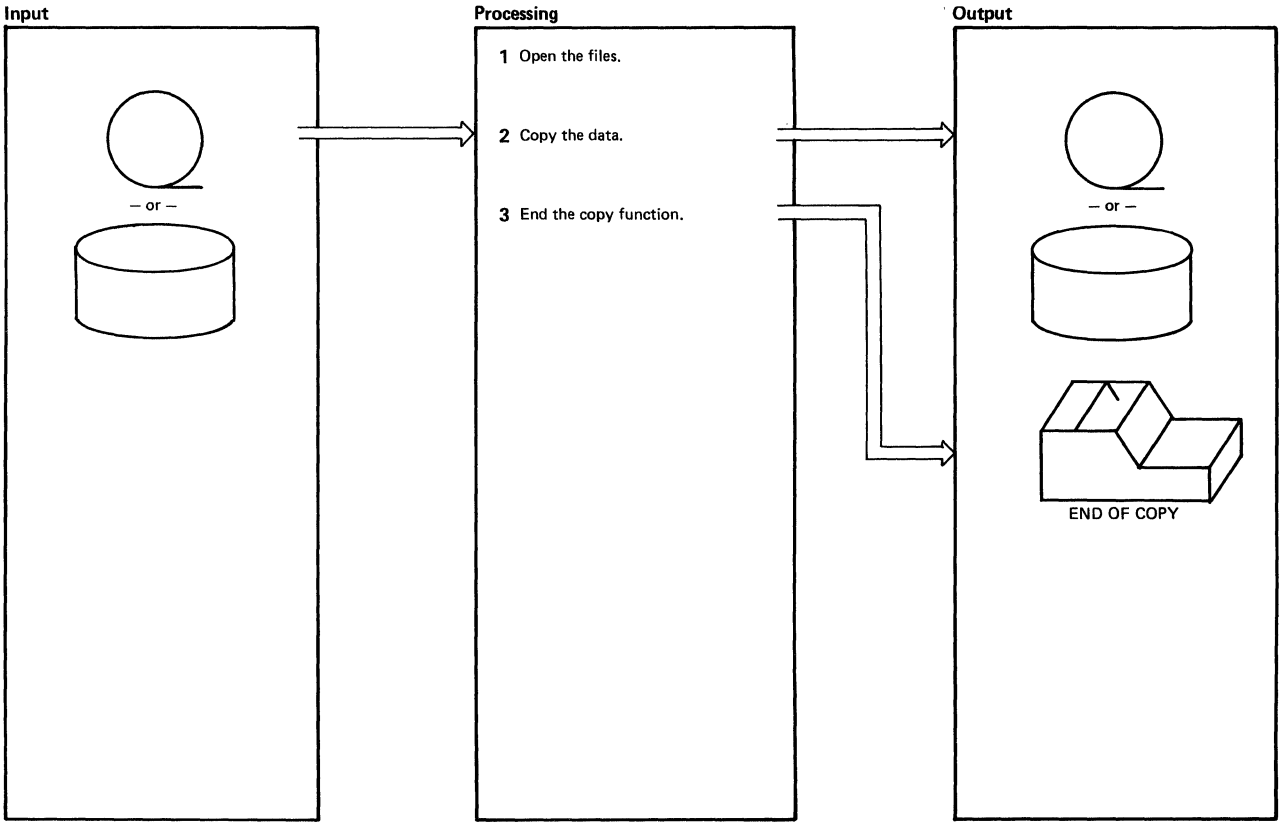
Notes		Module	Label	Ref
1	The work queues are initialized at OPEN time so that both buffers are on the tape input queue and both INIOB and OUTIOB are also in their respective queues. Therefore, QSEARCH selects TAPIN because the tape queue is searched first. TAPIN prepares to read the track header record tape block. It builds a CCW to read a 4K block and calls STARTIO (the unoverlapped entry point) to do the SIO. Because entry was to the unoverlapped entry of the start I/O routine, control is not returned until the I/O is complete. When the 4K tape record containing the THR record has been read, control returns to TAPIN. TAPIN Uses the control data in the THR to construct a CCW chain to read the rest of the 4K tape records, plus the last short record. TAPIN calls STARTIOO (the overlapped I/O entry point) to do the SIO. An SIO is issued. Because entry was to the overlapped entry point, control is returned when the condition code returned from the SIO is 0. The tape is now filing buffer A. TAPIN routes control directly to QSEARCH.	DMKDDR DMKDDR DMKDDR DMKDDR	QSEARCH TAPIN TAPIN STARTIOO	
	The remaining buffer (B) is still on the tape's input queue. However, the tape's IOB (INIOB) is not on the queue. Therefore, QSEARCH cannot select the tape input routine. Because no buffers are on the DASD or output queue, the output routine is also ineligible for selection. QSEARCH then loads an enabled wait PSW.	DMKDDR	QSEARCH	
	When the tape completes, the I/O interruption routine finds the owning IOB. In the IOB is the interruption return address (IOBIRA). When this routine is entered, register 15 equals the address of the IOB. TAPIRA now ensures that the FB-512 block number in question (INBLADD) is the data just read. If not, the TAPIN routine is reentered. If INBLADD is in the THR, the control fields in the THR are initialized in preparation for routing the buffer to the output routine. The fields are: THRWROTE - describes how many blocks in the buffer should be written THRFRSBL - the first block in the buffer. This block is left-justified in the buffer. THROBLAD - the block number on FB-512 where writing should begin. Next, control fields in the DDR are updated in preparation for the next input operation. These fields are: INBLADD - the block number of the next block to be read from tape OUTBLADD - the block number into which INBLADD should be written.	DMKDDR	TAPIRA	
2	If the blocks just read completed an extent, another field (CUREXT) is updated to point to the next entry in the extent table. If there are more blocks to process, return is to ENQIOB. If there are no more blocks to process, return to ENQBUF. This temporarily precludes further input.			
	The IOB address (R15) is stored in the appropriate queue. The INIOB address is sorted in the tape or input queue (Q1). Then the THR address (IOBTHR) is stored in the appropriate buffer list. The address of buffer A is stored in the output's list (B2) to signal that a buffer is ready to be output.	DMKDDR	ENQIOB	
	A channel program is built to read the next tape record. An SIO is issued and TAPIN builds the channel program to read the rest of the THR data. Unoverlapped SIO is issued.	DMKDDR	TAPIN	

Figure 27 (Part 1 of 2). Restore Function with Streaming

Notes		Module	Label	Ref
3	The tape input routine cannot be selected because the IOB is not on the queue. However, the fixed block output routine (FBAOUT) is selected to output buffer B. The IOB and buffer are dequeued and FBAOUT is entered.	DMKDDR	QSEARCH	
	Using the control data in the THR, a channel program is built to write the blocks. THROBLAD tells what block to LOCATE, and THRWROTE tells how many blocks to write. The overlapped entry to the start I/O routine is called. An SIO to DASD is issued. When cc=0, control is returned to the caller (FBAOUT). FBAOUT passes control to QSEARCH immediately.	DMKDDR	FBAOUT	
	Neither queue contains an available IOB address of buffer address. Both I/O devices are busy. An enabled wait PSW is loaded. Normally, the DASD finishes before the tape. The I/O interruption routine finds the owning IOB (OUTIOB in this case) and passes control to IOBIRA or FOUTIRA.	DMKDDR	QSEARCH	
	It is determined if the THR just ouput is the last one to be output in a given extent. If not, return is to ENQIOB. If it is, a call is made to PRINTEXT to print the extent map. If this is the last THR of the last extent, control passes to the EOJ routine. Otherwise, control is passed to ENQIOB. The IOB (OUTIOB) is enqueued to the appropriate queue (Q2), and the buffer (B) to the appropriate queue (Q1 the tape input queue). Control passes to QSEARCH.	DMKDDR DMKDDR	FOUTIRA ENQIOB	
	The tape is still busy (INTIOB is not on a queue) and there is no work for DASD output. A wait PSW is loaded. The tape interrupts after buffer B is read. IOBIRA is given control as before.	DMKDDR	QSEARCH	
4	Steps 1 through 3 are repeated. The cycle continues until all extents have been restored, or an EOV or EOJ label is read. If EOV, the I/O is interrupted while the tape switch occurs.			
	Then the cycles begin again with TAPIN building a channel program to read the next record. If EOJ is read the EOJ routine is called. In both cases, the output DASD is allowed to finish its queued work before tape processing (EOV or EOJ) is resumed.			

Figure 27 (Part 2 of 2). Restore Function with Streaming

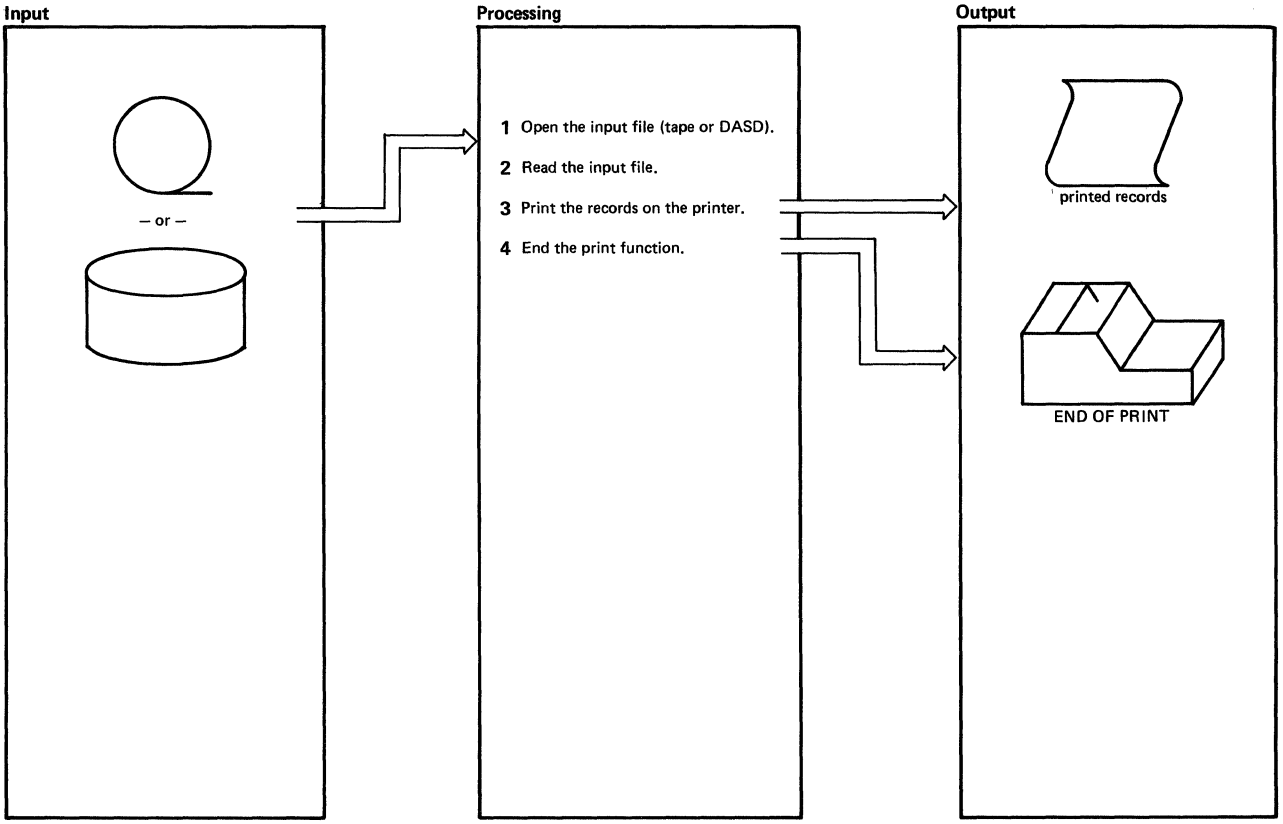
DASD Dump Restore Program



Notes		Module	Label	Ref
1	The input file and output file are opened. The input and output devices must be the same device type. It is allowable if both DASD devices are FB-512, regardless of the particular type. The extent table is updated to reflect the amount of data to be copied from one device to another.	DMKDDR	OPENIN GETEXT OPENOUT	
2	The heading is written and the message indicating the start of the copy function is typed.	DMKDDR	PRINTH MSG004	
	The input file is read and the output file is written. If copying from disk to disk, the pointers to the disk records are updated to the next record. The read write cycle continues until the specified data is copied. When copying data from tape to tape, the GETTHR routine performs the record read and the TESTOUT routine performs the record write. When copying data from disk to disk, the BUILDTHR routine performs the record read and the DASDWRIT routine performs the record write.		UPDTADD GETTHR TESTOUT BUILDTHR DASDWRIT SCANCOPY	
	If the COMPACT option was specified, the following message is displayed: DMKDDR731I COMPACT OPTION IGNORED FOR COPY OPERATIONS		DCMP500	
3	The message END OF COPY indicates the successful completion of the copy function.	DMKDDR	CLOSEJOB	
	When copying data from tape to tape, the output tape is positioned as indicated on the COPY control card. When the disk to disk copy is complete, the disk is closed.			
	Control returns to the GTCARD routine to read the next control statement.			

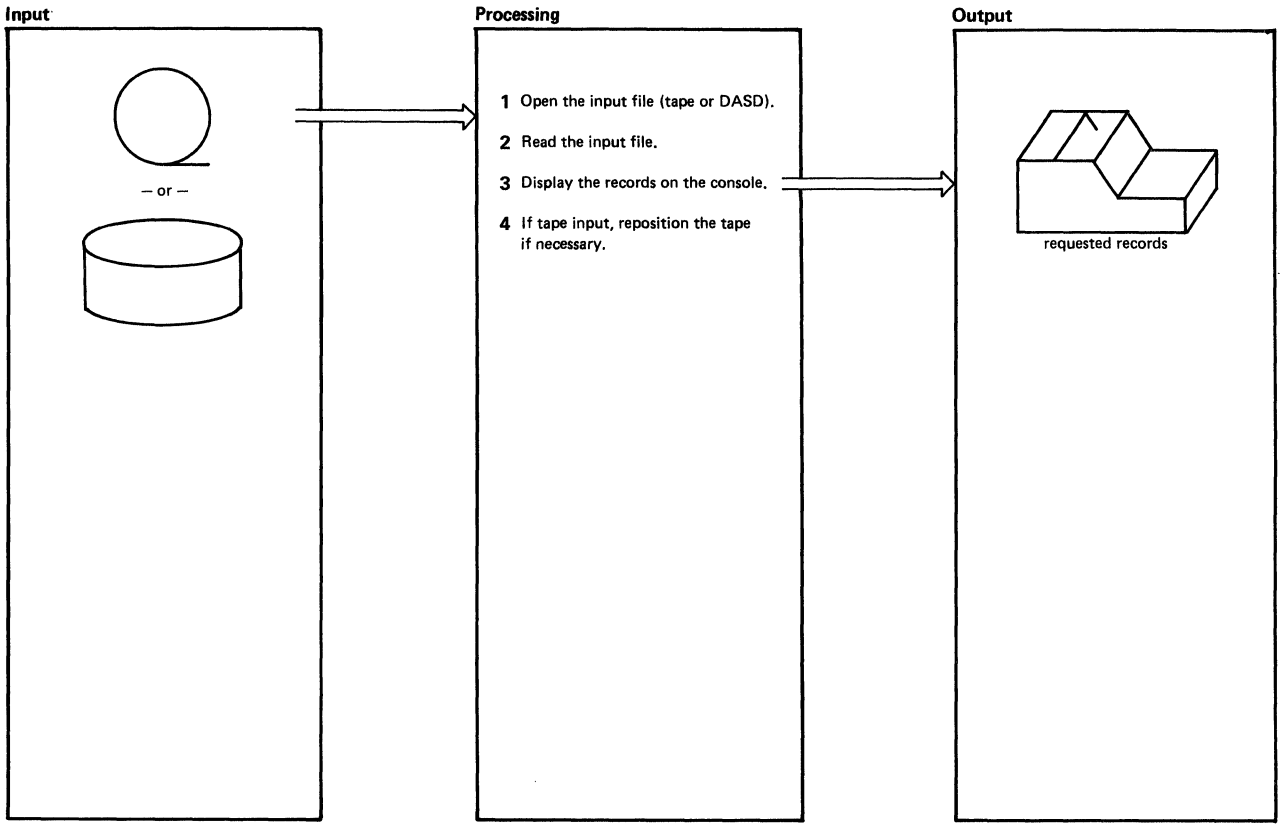
Figure 28. Copy Function

DASD Dump Restore Program



Notes		Module	Label	Ref
1	The input device is opened. If the input is on tape, the tape is spaced forward the designated number of records (if any). The extent table is updated to reflect the cylinders to be printed.	DMKDDR	OPENIN GETEXT	
2	The message PRINTING xxxxxxxx is displayed to indicate the start of the PRINT function.	DMKDDR	MSG004	
3	The data is read from the input device via the appropriate (disk or tape) read routine. The data is converted, decoded from compact format if needed, and printed on the system printer.	DMKDDC DMKDDT DMKDDR	GETTHR DCMP500 DISPLAY	
4	The message END OF PRINT indicates the successful completion of the PRINT function.	DMKDDR	EOJ	
5	Control returns to the GTCARD routine to read the next control statement.			

Figure 29. Print Function



Notes		Module	Label	Ref
1	The input device (either tape or disk) is opened. If input is on tape, the tape is spaced forward the designated number of records (if any). The extent table is updated to reflect the data to be typed.	DMKDDR	OPENIN GETEXT	
2	The records are read from the tape or disk by the appropriate read routine and decoded from compact format, if needed.	DMKDDR DMKDDC DMKDDT	BUILDTHR GETTHR DCMP500	
3	The records are displayed on the console. The read and type cycle is continued until all the specified records are typed.	DMKDDR	DISPLAY	
4	Control returns to the GTCARD routine to read the next control statement.	DMKDDR	EOJ	

Figure 30. Type Function

Program Organization

This section contains a program description of the DMKDDR module.

DMKDDR

The DASD dump restore program.

Attributes

Serially reusable.

Entry Point

DMKDDREP.

Registers at Entry

Reg	Use
R1	Points to a parameter list when DMKDDR is executed under the control of CMS.

Registers at Exit

Reg	Use
R15	Contains a return code when DMKDDR is executed under the control of CMS. The return codes are:

Return Code	Meaning
1	Invalid filename or file not found
2	Error while running the program
3	Flagged DASD track
4	Permanent tape or DASD I/O error
20	Error in the decoding routine
1xx	Error in the PRINTIO routine
2xx	Error in the CONREAD routine
3xx	Error in the RDBUF routine
4xx	Error in the TYPLIN routine

where: xx is the return code from the CMS routine.

Register Usage

Reg	Use
R0	Work Register.
R1	Pointer to input field from SCANCONT. Pointer to the output buffer (PRINT/TYPE). Work register.
R2	Input count from SCANCONT. Unit address for STARTIO. Data block count (PRINT/TYPE). Work register.
R3	End of current line (PRINT/TYPE). Work register.
R4	Length of one line (PRINT/TYPE). Pointer to key (PRINT/TYPE). Work register.
R5	Total length of data (PRINT/TYPE). Work register.
R6	Data count (PRINT/TYPE). Number of records on the track (PRINT/TYPE). Work register.
R7	Pointer to the extent table entry. Current line pointer (PRINT/TYPE).
R8	Address of the data area used for DASD/tape input and output (THR).
R9	Base register 5.
R10	Base register 1.
R11	Base register 2.
R12	Base register 3.
R13	Base register 4.
R14	Return address
R15	Pointer to the IOB.

External References

DMSACF, DMSCRD, DMSCWR, DMKDDC (data decoding,) DMKDDT (decoding table,) DMKDNC (data encoding,) DMKDNT (encoding table)

Directory

Following is an alphabetic list of the major labels in the DASD Dump Restore program. The associated method of operation diagrams are indicated and a brief description is included of the operation performed at the point in the program that is associated with each label.

Label	Figure	Description
ADDLINE		Checks for duplicate line.
ALL		Handles the ALL parameter.
ALLSET		Prepares to type or print data specified.
ALTTRACK		After errors, handles alternate tracks.
ALTXDEF		Handles defective alternate tracks.
BINCONV		Converts decimal numbers to binary.
BLDFBATH		Builds THR for FBA devices.
BLNKVSER		Clears the volume serial number on disk.
BSFILE		Backspaces if overran VHR on tape.
BUILDCCW		Builds a CCW string to put the key/data fields into the THR (track header record.)
BUILDIOB		Creates the IOBLOK for DDR.
BUILDTHR	24 28 29 30	Reads records from disk.
CCMP200	24 26	Subroutine prints encoding/decoding statistics.
CCMP300	24	Subroutine encodes data into compact format before writing to tape.
CHKCOPY		Initiates double buffering for the copy function.
CHKEOE	25	Checks if output was for last of an extent. If yes, prints the extent map.
CHKLOW		Insures nucleus starts in permanent space for FBA.
CHKSAME		Checks for same device type.
CHKSAM1		Checks for same FBA device type.
CHKSIZE		Checks incorrect length errors.
CHKTORE		Initiates double buffering for the restore function.
CHKTYPE		Sees if user and CP agree.
CKDEOV		Handles EOV conditions for CKD devices.
CKDVHRIN		Initializes VHR for CKD devices.
CKDVOL		Reads volid from DASD.
CKEXT		Checks for ECKD.
CLOSEJOB	24 26 28	Displays message indicating the end of a DDR function.
CLOSE1		Closes the tape and reads another.
CLOSIT		Ends a job step.
CLOSJ		Displays message indicating the end of a DDR function.
CMPCPY		Skips if COMPACT not specified for the copy function
CMPPEND		Bypasses statistics for encoding routines.
CMSA		Handles console output.

Label	Figure	Description
CMS1	22	Builds a PLIST (parameter list) if parameters passed from CMS.
CMS8	22	The end-of-job processing when DDR is running under VM/SP.
COMPARE		Compares keywords.
COMPBLK		Checks the addresses for FBA devices.
COMPCYL		Checks the home addresses for CKD devices.
COMPSIZ		Compares relative sizes of devices for the copy function.
CONERROR		Handles console errors.
CONRET		Returns here after console wait.
CONSOUT2		Handles sysprint console output.
CONTSCAN		Gets extent table for the type and print functions.
CORRCSW		Handles imprecise ending conditions.
CPVOL		Handles all active DRCT and PERM space.
CSWSTORE		Checks CSW after a SIO.
CYLSETOK		Sets up output cylinder id.
DASDRCVR		Gets the address of the CCW chain to write to DASD.
DASDWRIT	26 28	Writes records onto disk.
DATACHK1		Checks for read multiple CKD.
DATACHK2		Checks for permanent errors.
DATACOR1		Corrects ECC correctable errors and restarts the chain.
DCMP500	26 28 29 30	Subroutine invokes decompacting during reads.
DDR536		Issues DMKDDR536I message.
DDR700		Issues DMKDDR700E message.
DDR701		Issues DMKDDR701E message.
DDR702		Issues DMKDDR702E message.
DDR703		Issues DMKDDR703E message.
DDR704		Issues DMKDDR704E message.
DDR705		Issues DMKDDR705E message.
DDR707		Issues DMKDDR707E message.
DDR708		Issues DMKDDR708E message.
DDR709		Issues DMKDDR709E message.
DDR710		Issues DMKDDR710A message.
DDR711		Issues DMKDDR711R message.
DDR712		Issues DMKDDR712E message.
DDR713		Issues DMKDDR713E message.
DDR714		Issues DMKDDR714E message.
DDR715		Issues DMKDDR715E message.
DDR716		Issues DMKDDR716A message.
DDR717		Issues DMKDDR717R message.
DDR718		Issues DMKDDR718E message.
DDR719		Issues DMKDDR719E message.
DDR720		Issues DMKDDR720E message.
DDR721		Issues DMKDDR721E message.
DDR722		Issues DMKDDR722E message.
DDR723		Issues DMKDDR723E message.
DDR724		Issues DMKDDR724E message.
DDR725		Issues DMKDDR725R message.
DDR726		Issues DMKDDR726E message.
DDR727		Issues DMKDDR727E message.
DDR728		Issues DMKDDR728E message.
DDR729		Issues DMKDDR729I message.

Label	Figure	Description
DDR731		Issues DMKDDR731I message.
DDR756		Issues DMKDDR756E message.
DECCONV		Converts decimal numbers to hexadecimal.
DEFTRACK		Handles defective tracks.
DEVICOK		Device is waiting, start the I/O.
DEVTEST		Tests the device types for the dump function.
DEVTST		Tests the device types for the restore function.
DISPFT3		Handles FTR format before the display function.
DISPIT		Displays the key/data message.
DISPLAY	29	Prints or types records.
	30	
DISPR0		Prints record 0.
DMKDDR		Start of the DMKDDR module.
DMKDDREP	22	Entry point to the DDR program.
DODIV		For FBA, finds the number of blocks/track.
DOTAPIO		Issues SIO to write to tape.
DOTAPIO2		Reads in the FTR records.
ENQIOB	25	Moves a ready buffer to the queue for processing.
	27	
EOJ	24	At the end of a DDR function, returns control to the
	29	GTCARD routine.
	30	
ERRCLOSE		Closes tape and reads alternate tape.
EXIT	22	Returns to CMS command environment or enters wait state
EXTENTIN		Gets the cylinder extents for DASD.
EXTINT		Handles external interrupts.
FADD1		For FBA, converts allocation map to blocks.
FBACPVOL		Reads in the allocation extent map.
FBADISP		Displays FBA blocks.
FBAEOV		Handles EOV conditions for FBA devices.
FBAERR		Checks FBA device errors.
FBAEXTS		Builds the extent table for FBAs.
FBAIN	25	Calls STARTIOO to read the FB-512 device.
FBALLC		Handles nucleus formatting on FBA devices.
FBAOUT	27	Calls STARTIOO to write for FB-512 device.
FBARDCIO		RDC command - gets 32 bytes of device characteristics.
FBASTOP		Gets stop block for the type and print functions.
FBAUPADD		Updates pointers to next set of FBA blocks.
FBAWRIT		Writes blocks to FBA volumes.
FINIRA	25	Calls CHKEOE to see if an extent has just been finished.
FOUNDIT		Finds the error address.
FOUTIRA	27	Updates for the next input operation.
FTRDOK		Prepares to write out the THR.
FTREAD		Reads in FTR records for DASD.
GETCCHH		Returns CCHH address from sense data.
GETCPTYP		Sees if user and CP agree about device type.
GETCSW		Picks up the error CSW command address.
GETDASD		Sets up for cpvol nucleus dump.
GETEXT	24	Builds extent table.
	26	
	28	
	29	
	30	

Label	Figure	Description
GETFTREC		Reads the full track records from tape.
GETHAR0		Reads home address and record 6.
GETNEWEX		Gets new extent for FBA.
GETOTHER		Scans for other options on DASD.
GETPARM		Handles tape options.
GETREOR		Tests for reorder parameter for CKD.
GETREOR1		Tests for reorder parameter for FBA.
GETR1		Checks for records that need to be printed.
GETSTART		Gets the next statement.
GETTHR	26	Reads tape records.
	28	
	29	
	30	
GETVHR		For tape input, gets VHR.
GETVSER		Opens the DASD unit.
GOODNAME		Finds a valid name and returns.
GOSUB1		Gets the next record.
GOTTHR		Finds the THR.
GRAPHID		Handles I/O for display terminals.
GTCARD	22	Reads control cards.
GTSCAN		Gets the first field on the card.
HEADOK		Fills in header record fields.
HEXCONV		Converts hexadecimal numbers to decimal.
INOUTER		Handles tape and DASD errors.
IOERROR		Handles I/O errors.
IOWAIT		Enables for I/O interruptions.
LASTONE		Checks for last record.
LOOP5		Scans control statements for next field.
LOOP12		Checks for last record to be displayed.
LOOP13		Determines the starting address.
MARKOPEN		Marks the IOB as open for the device.
MSGRET		After message is written, scans the next line.
MSGWRITE		Displays messages on the terminal.
MSG001		Writes 'END OF xxxxx' message.
MSG002		Writes header message.
MSG003		Writes 'ENTER EXTENTS' message.
MSG004	24	Prints message indicating start of Dump, Restore, Copy,
	26	or Print function.
	28	
	29	
MSG005		Writes 'MOUNT NEXT TAPE' MESSAGE.
MSG006		Writes 'MOUNT NEXT TAPE' MESSAGE.
NEWADD	22	Prints heading when DDR program running standalone.
NEXTCYL		Updates pointer to next cylinder.
NEXTREC		Updates pointer to next record.
NEXTTCK		Updates pointer to next track.
NORECFND		Handles a 'NO RECORD FOUND' check.
NOSTART		Sets up starting address for DMKDDR721E message.
NOTCONS1		Ignores CONS option if not under CMS.
NOTFTRD		Does normal read processing.
OK		Points to read CCWs to read THR.
OPENCNT		Handles streaming for tapes.
OPENDASD		Opens a DASD.
OPENER		Handles user responses.

Label	Figure	Description
OPENIN	24 26 28 29 30	Opens input devices.
OPENOUT	24 26 28	Opens output devices.
PBUFFER		Points to the print buffer.
PCOUNT		Converts all addresses and data to decimal.
PDATA		Sets up print pointer.
PRINTBUF		Prepares to write out error message.
PRINTDAT		Prints the data.
PRINTER1		Updates the printer line count.
PRINTER2		Spaces the printer twice.
PRINTEXT		Handles printer output.
PRINTH	24 28	Prints function heading.
PRINTIT		Prints the header message.
PRINT1		Checks that device type is console.
PRINT2		Displays message on console.
PRTINIT		Initializes the printer.
PUBLKUP		Sees if there is a device waiting.
QSEARCH	25 27	Looks for work on queues. If found, dequeues IOB and a buffer and enters I/O routine.
QUIESCE		Handles queued work for double buffering.
READCKP		Reads nucleus.
READCONT		Reads control statements.
READCT		Reads the home address, record 0, and the count fields.
READKEYD		Reads the key and data records.
READTAPE		Reads in the records.
READ66		Reads data from graphics devices.
REORBLOK		Reorders FBA blocks for output.
REORCYL		Writes out the THR.
REPOTAPE		Repositions the tape after an error.
RESPONSE		Handles user responses from DDR questions.
RESTART		Restarts the I/O.
RETRY		Sets up for retry of the SIO.
RSPCPY		Issues responses for the copy function.
SAVECT		Saves the printer line count.
SCANCONT		Scans control statements for next operand.
SCANCOPY	23 28	Scans the COPY function statement.
SCANDATA		Scans control statements for special characters.
SCANDUMP	23	Scans the DUMP function statement.
SCANFBA		Scans for starting and ending blocks for FBA type and print functions.
SCANFTR		Scans for FULL TRACK READ option.
SCANINPU	23	Scans the INPUT control statement.
SCANLEAV		Scans for LEAVE option.
SCANMODE		Scans for MODE option.
SCANNAME		Scans the name table (TABLE1) for a matching control statement name.

Label	Figure	Description
SCANOUTP	23	Scans the OUTPUT control statement.
SCANPRIN	23	Scans the PRINT function statement.
SCANREST	23	Scans the RESTORE function statement.
SCANSKIP		Scans for SKIP option.
SCANSYSP	23	Scans the SYSPRINT control statement.
SCANTYPE	23	Scans the TYPE function statement.
SCANUNIT		Scans the device table (TABLE2).
SCANUNLO		Scans for UNLOAD option.
SCRATCH		For scratch volser, skips label verification.
SENSIO		Does a sense on the device.
SETDASD	26	Checks volume serial number of output disk.
SETEND		Prints the cylinder map at end-of-job.
SETEXT		Picks up the cylinder number that starts the next extent.
SETEXT1		Picks up the block number that starts the next extent.
SETMK		Builds CCW chain for reading the records.
SETQ		Initiates double buffering.
SETSTOP		Gets stop cylinder for the type and print functions.
SETUPADD		Sets up input and output addresses.
SETUPBUF		Clears the print buffer.
SETUPERR		Handles errors writing to the console.
SETVSN		Sets up volume serial number.
SET4K		Reads in Non-FTR records.
SKIPMSG		Prints record overflow message.
STARTIO		Starts I/O devices.
STARTIOO	25 27	Issues SIO.
STMSHRT1		Issues CCW for writing short records.
STOREADD		Starts here after the first read.
SUPMSG		Prints the suppress line message.
TABLE1		Generated name/function table.
TABLE2		Generated unit/device table.
TAPE		Checks for an alternate tape device address.
TAPEER		Checks tape device errors.
TAPIN	27	Reads the THR and calls STARTIOO to read the remaining tape records.
TAPIRA	27	Updates DDR in preparation for the next input operation.
TAPNCMP		Checks for FTR output format.
TAPOUT	25	Writes THR to tape.
TAPWRIT		Prepares to write to the tape.
TESTALLF		Tests for the 'RESTORE ALL' function.
TESTCARD		Checks for card input at end-of-job.
TESTCMS	22	Exits by entering wait state when DDR program is running standalone.
TESTCOMR		Tests for command reject.
TESTDACK		Tests for data checks.
TESTDASD		Tests for DASD in printer routine.
TESTDEV		Tests device status after SIO.
TESTEND		Terminates when blank card read.
TESTFLAG		Checks for NUCLEUS option.
TESTGRAP		Tests for a graphics device.
TESTIN		Checks for tape input.
TESTINF		Checks input block numbers.
TESTIO		Does a TEST I/O on the device.
TESTMD		Tests for FTR errors.

Label	Figure	Description
TESTNPAG		Skips printer to channel 1.
TESTOPT		Handles the type and print function options after left parenthesis.
TESTOUT	24 28	Writes tape output records.
TESTPERM		Tests for permanent allocated space.
TEST800		Tests for 800 MODE option.
TEST1600		Tests for 1600 MODE option.
TEST3278		Tests for a 3278 device.
TES3270T		Tests for a 3270 device.
TPSWP		Closes the old tape and opens the next tape.
TRANS		Translates data to printable characters.
TRKCOND		Recovery procedure for track condition check (alternate track).
TSTCOUNT		Prints the end of the track.
TSTDEV		Tests the device types for the copy function.
TSTEXT		Tests the extents for output devices.
TSTINPUT		Common code for the type and print functions when opening device.
UNITCHK		Handles unit checks.
UPDTADD	24 26 28	Updates disk addresses.
UPDTEXT		Restores entire track.
UPDTEXT1		Restores blocks for FBA devices.
USENBLK		Initializes THR fields.
VALEXT		Validates the cylinder or block number.
WCKDSET		Sets up extended CCWS.
WDSIO		Writes the THR (track header record).
WRITENUC		Handles transferring of nucleus.
WRTSFMT		Handles overflow records.
WRT66		Writes data to graphics devices.
WTDASD		Writes THR to DASD.
YEARSET		Calculates the Greenwich Mean Time.

Data Areas

This section contains a description of a:

- DDR Trace Table
- Cylinder header record
- Track header record
- IOB.

Trace Table

Figure 31 shows the trace table format. Trace table addresses may be obtained by referencing the module and finding these labels:

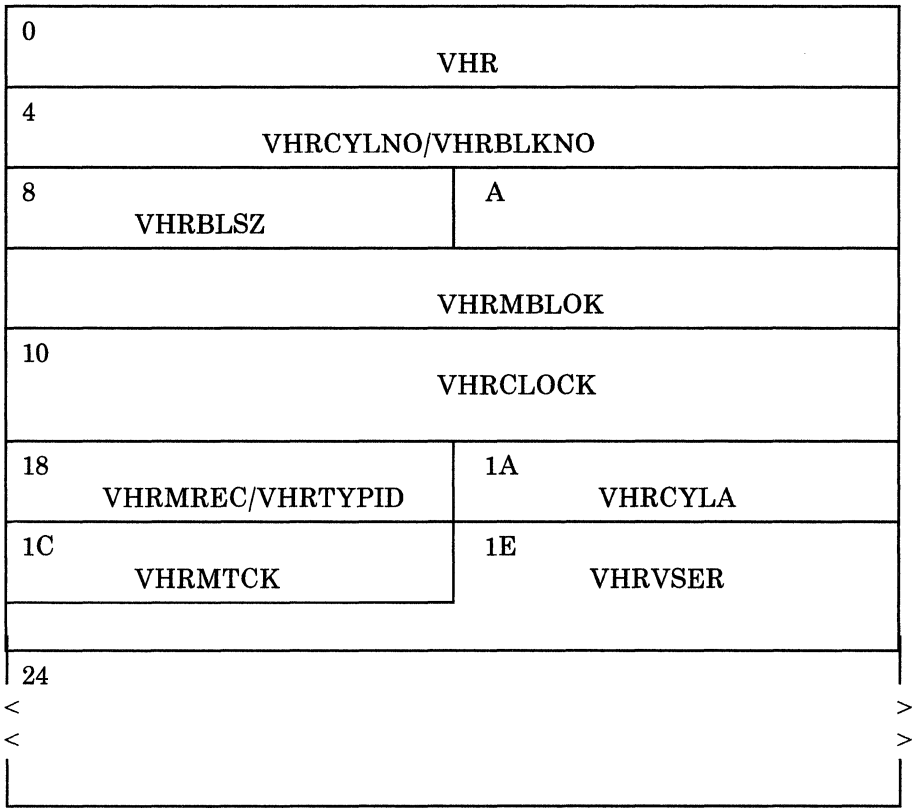
- TRACEST - beginning of trace table
- TRACEND - end of trace table
- TRACEPT - pointer to next available entry.

Event	Id. Code	Format of Trace Entry
Start I/O IPL	E2	X'E2' | CAW | First IOB word | Return address | IOB (1 | 4 | 8 | 12)
Interrupt [1] (native)	C9	X'C9' | I/O Old PSW | CSW
Error	C5	X'C5' | CAW | Davice address | Sense information
Interrupt (virtual)	C9	X'C9' | CAW or sense | Diagnose 20 RC | CSW

[1] Byte 0 is overlayed by the trace indentification code.

Figure 31. DDR Trace Table (internal)

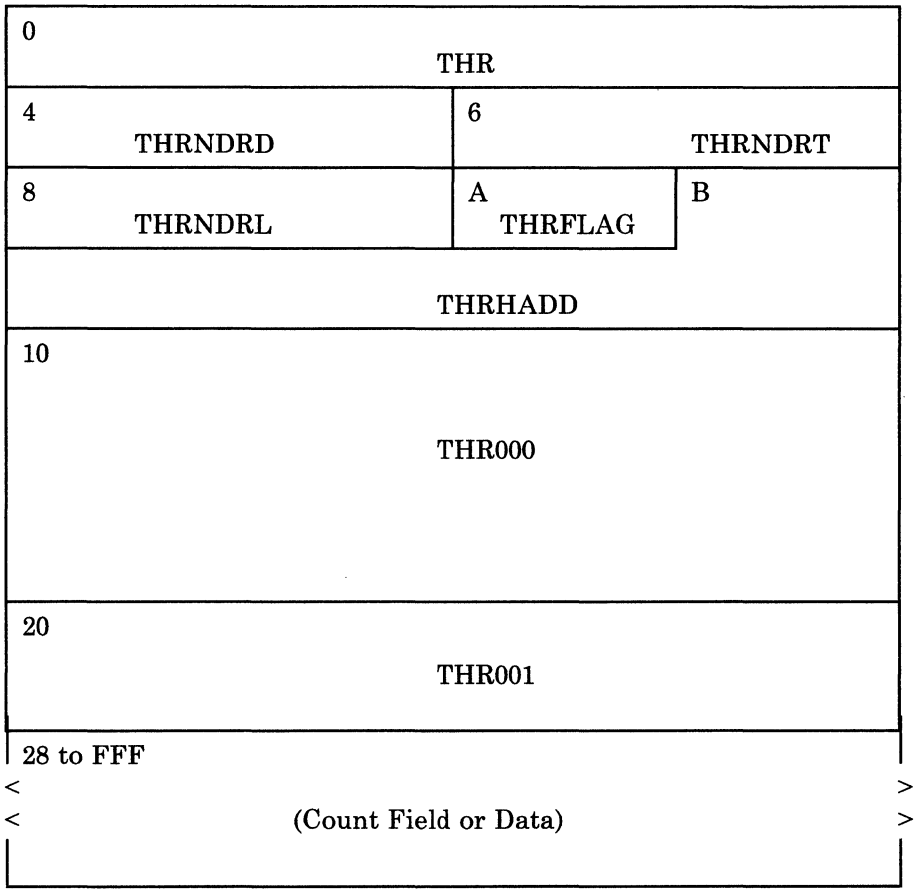
Cylinder Header Record



Displacement Hex	Dec	Field Name			Description
0	0	VHR	DC	CL4'VHR'	
4	4	VHRCYLNO	DS	CL6'0'	BBCCHH of input DASD unit
4	4	VHRBLKNO	DS	F'0'	FB-512 block number of first block
8	8	VHRBLSZ	DS	H	Block size of originating device
			DS	H	Not used (for alignment)
A	10		DS	XL6' '	Not used if count-key-data
C	12	VHRMBLOK	DS	F	Maximum block number of originating device
10	16	VHRCLOCK	DS	D'0'	Time of day clock value
18	24	VHRTYPID	DS	H'0'	Halfword of zeros identifies that this volume contains FB-512 data
1A	26	VHRCYLA	DS	H'0'	CC address of last cylinder on this type of DASD
1C	28	VHRMTCK	DS	H'0'	
1E	30	VHRVSER	DS	CL6'VOLSER'	Volume serial number of input DASD unit
24	36		DS	CL44' '	

Figure 32. Cylinder Header Record

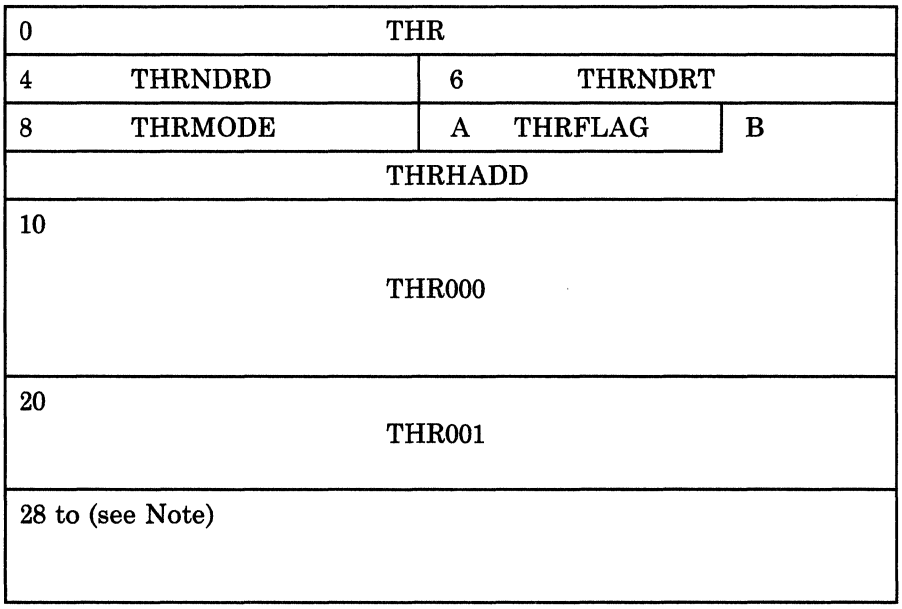
Track Header Record for Count-Key-Data (non-FTR)



Displacement Hex	Dec	Field Name		Description
0	0	THR	DC CL4'THR '	ID of track header record
4	4	THRNDRD	DC H'0'	The number of count fields in the THR
6	6	THRNDRT	DC H'0'	The number of 4K data records on tape
8	8	THRDRL	DC H'0'	Length of the short (last) data record
A	10	THRFLAG	DC XL1'0'	Flag
		Bit Settings for THRFLAG:		
		SPECIAL	EQU X'01'	Overflow
B	11	THRHADD	DC XL5'0'	The home address reordered
10	16	THR000	DC XL16'0'	Record 0 from the DASD unit
20	32	THR001	DC XL8'0'	Count field of the first record
28	40			Count fields and data

Figure 33. Track Header Record for Count-Key-Data (non-FTR)

Track Header Record For Count-Key Data (FTR)

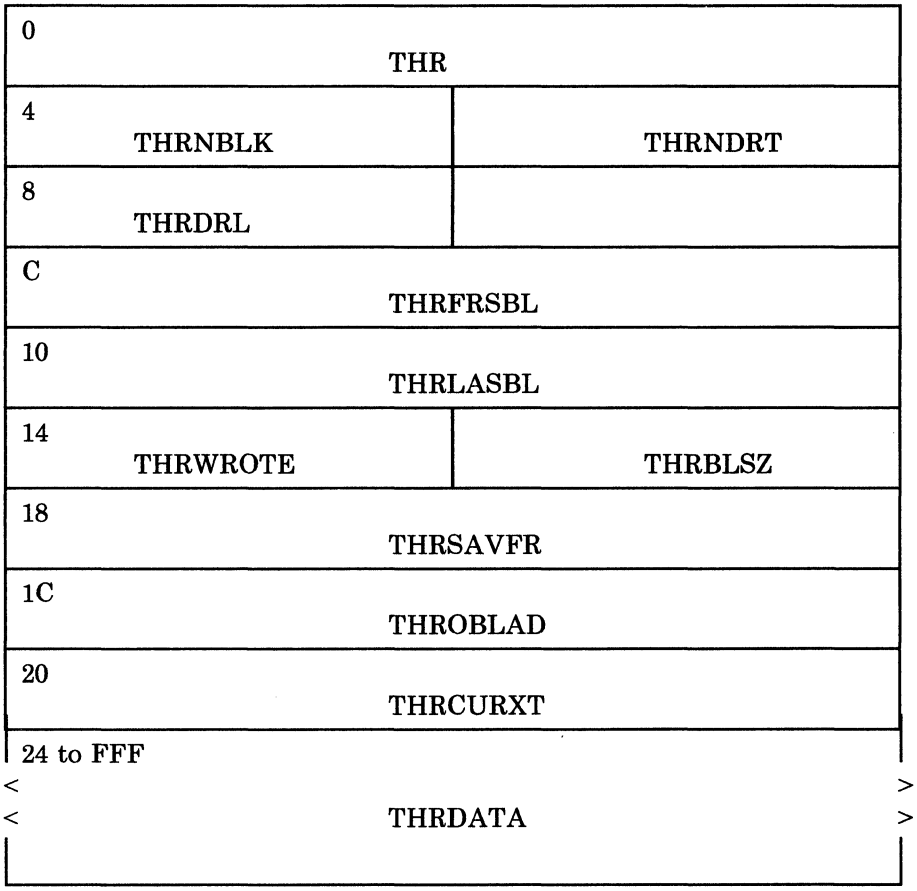


Displacement Hex	Dec	Field Name		Description
0	0	THR	DC CL4'THR '	ID of track header record
4	4	THRNDRD	DC H'0'	The number of records in the track
6	6	THRNDRT	DC H'0'	The track length in bytes
8	8	THRMODE	DC XL1'0'	bpi setting for tape
		Bit settings for THRMODE:		
		MODE6250	EQU X'00'	For 6250 bpi tape
		MODE1600	EQU X'01'	For 1600 bpi tape
		MODE800	EQU X'02'	For 800 bpi tape
		MODE38K	EQU X'00'	For 38K bpi tape
A	10	THRFLAG	DC XL1'0'	Flag for track status
		Bit settings for THRFLAG:		
		SPECIAL	EQU X'01'	Special
		FTRMODE	EQU X'02'	Header in FTR mode
B	11	THRHADD	DC XL5'0'	The home address reordered
10	16	THR000	DC XL16'0'	Record 0 from the DASD unit
20	32	THR001	DC XL8'0'	Count fields of the first record
28	40			Count-key-data fields

Note: 28 to 1FFF for 800 bpi, 28 to 2FFF for 1600 bpi, 28 to BFFF for 6250 bpi, and 38K bpi tape.

Figure 34. Track Header Record for Count-Key-Data (FTR)

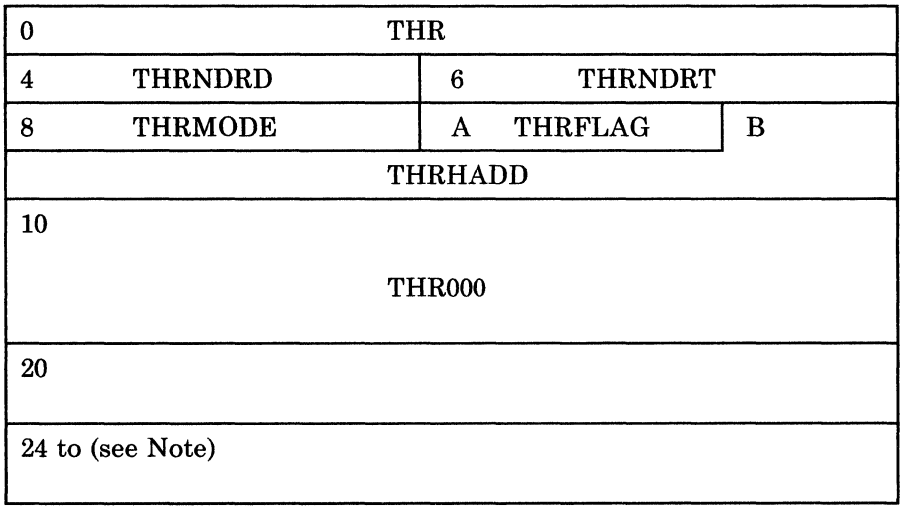
Track Header Record for FB-512



Displacement Hex	Dec	Field Name			Description
0	0	THR	DC	CL4'THR'	ID of track header record
4	4	THRNBLK	DS	H	The number blocks in the record
6	6	THRNDRT	DC	H	The number of 4K data records on tape
8	8	THRDRL	DC	H	Length of the short (last) data record on tape
A	10		DS	X	Reserved for IBM use
B	11		DS	X	Reserved for IBM use
C	12	THRFRSBL	DS	F	First block in record
10	16	THRLASBL	DS	F	Last block in record
14	20	THRWROTE	DS	H	Number of blocks to be output
16	22	THRBLSZ	DS	H	512 - the size of one block
18	24	THRSAVFR	DS	F	Save area for first block number
1C	28	THROBLAD	DS	F	Block number where output should begin
20	32	THRCURXT	DS	F	Address of entry in extent
24	36	THRDATA			The actual FB-512 data

Figure 35. Track Header Record for FB-512

Track Header Record For Count-Key Data (Compacted, FTR or Non-FTR)



Displacement Hex	Dec	Field Name		Description
0	0	THR	DC CL4'THR'	ID of track header record
4	4	THRNDRD	DC H'0'	The number of records in the track
6	6	THRNDRT	DC H'0'	The number of compacted bytes
8	8	THRMODE	DC XL1'0'	bpi setting for tape
		Bit settings for THRMODE:		
		MODE6250	EQU X'00'	For 6250 bpi tape
		MODE1600	EQU X'01'	For 1600 bpi tape
		MODE800	EQU X'02'	For 800 bpi tape
		MODE38K	EQU X'00'	For 38K bpi tape
A	10	THRFLAG	DC XL1'0'	Flag for track status
		Bit settings for THRFLAG:		
		SPECIAL	EQU X'01'	Special
		FTRMODE	EQU X'02'	Header in FTR mode
		CMPCOMP	EQU X'80'	Encoded data
		CMPLBLK	EQU X'40'	Compaction flag
B	11	THRHADD	DC XL5'0'	The home address reordered
10	16	THR000	DC XL16'0'	Record 0 from the DASD unit
20	32		DS F	Number of bytes of non-compacted data
24	36			Compacted data

Note: 24 to 1FFF for 800 bpi, 24 to 2FFF for 1600 bpi, and 24 to BFFF for 6250 bpi and 38K bpi tape.

Figure 36. Track Header Record for Count-Key-Data (Compacted, FTR or Non-FTR)

Track Header Record for FB-512 (Compacted)

0	THR	
4	THRNBLK	THRNDRT
8	THRDRL	
C	THRFRSBL	
10	THRLASBL	
14	THRWROTE	THRBLSZ
18	THRSAVFR	
1C	THROBLAD	
20	THRCURXT	
24		
28 to (see Note)		

Displacement Hex	Dec	Field Name			Description
0	0	THR	DC	CL4'THR'	ID of track header record
4	4	THRNBLK	DS	H	The number of blocks in the record
6	6	THRNDRT	DC	H	The number of compacted bytes
8	8	THRDRL	DC	H	Length of the short (last) data record on tape
A	10	THRFLAG	DC	XL1'0'	Track Header Record flag
		Bit settings for THRFLAG:			
		SPECIAL	EQU	X'01'	Special
		FTRMODE	EQU	X'02'	Header in FTR mode
		CMPCOMP	EQU	X'80'	Encoded data
		CMPLBLK	EQU	X'40'	Compaction flag
B	11		DS	X	Reserved for IBM use
C	12	THRFRSBL	DS	F	First block in record
10	16	THRLASBL	DS	F	Last block in record
14	20	THRWROTE	DS	H	Number of blocks to be output
16	22	THRBLSZ	DS	H	512 - the size of one block
18	24	THRSAVFR	DS	F	Save area for first block number
1C	28	THROBLAD	DS	F	Block number where output should begin
20	32	THRCURXT	DS	F	Address of entry in extent
24	36		DS	F	Number of bytes of non-compacted data
28	40				Compacted data

Note: 28 to 1FFF for 800 bpi, 28 to 2FFF for 1600 bpi, and 28 to BFFF for 6250 bpi and 38K bpi tape.

Figure 37. Track Header Record for FB-512 (Compacted)

IOB

0 IOBSTAT	1 IOBOPT	2 IOBUADD
4 IOBCCW		
8 IOBERROR		
C IOBCSW		
14 IOBCLASS	15 IOBTYPE	16 IOBMREC
18 IOBCYLP		1A IOBCYLA
1C IOBMTCK		1E IOBMODE / 1F IOBDISP
20 IOBVSER		
24		26 IOBATAPE
28 IOBFLAG	29 Reserved for IBM use	
2C IOBWHATQ		
30 IOBTHR		
34 IOBIRA		
38 IOBOUTBF		

Displacement Hex	Dec	Field Name		Description
0	0	IOBSTAT	DS X'80'	Status of IOB
		Bit settings for IOBSTAT:		
		IOBST	EQU X'80'	I/O unit is to be started
		IOBSTACK	EQU X'40'	I/O error has been stacked
		IOBLAST	EQU X'20'	Last IOB
		IOBNOPER	EQU X'10'	Device is not operational
		IOBCPVOL	EQU X'08'	Unit is a CPVOL
		IOBOPEN	EQU X'04'	The IOB is open
		IOBSCRAT	EQU X'02'	The DASD device is a scratch volume
		IOBTPSWP	EQU X'01'	Switch to alternate tape in progress
1	1	IOBOPT	DS 1X	IOB flags
		Bit settings for IOBOPT:		
		IOBDEW	EQU X'80'	Wait for device end interrupt
		IOBERST	EQU X'40'	Stop on I/O error and wait for next interrupt
		IOBEEXIT	EQU X'20'	Repeat CCW on error
		IOBSIO	EQU X'10'	Do not use Diagnose I/O
		IOBOVER	EQU X'08'	Used only by SIO routine. Means entry was via STARTIO overlay entry; therefore, do SIO and return to caller when subsequent cc=0
		IOBFTR	EQU X'02'	Use full track read feature
2	2	IOBUADD	DS 1H	Unit address of device
4	4	IOBCCW	DS 1F	Pointer to CCW
8	8	IOBERROR	DS A	Address of IO error routine
C	12	IOBCSW	DS 2F	CSW of IO error stacked
14	20	IOBCLASS	DS X'0'	Device class
15	21	IOBTYPE	DS X'0'	Device type
16	22	IOBSKIP	EQU *	IOB type skip count
16	22	IOBMREC	DS H'0'	Maximum number of records that will fit a track
18	24	IOBCYLP	DS H'0'	Maximum primary cylinder address of DASD device
1A	26	IOBCYLA	DS H'0'	Maximum alternate cylinder address of DASD device
1C	28	IOBMTCK	DS H'0'	Maximum number of tracks (numbering 0-N)
1E	30	IOBMODE	DS X	IOB tape mode command code
1F	31	IOBDISP	DS X	IOB tape disposition command code
20	32	IOBVSER	DS CL6' '	Volume serial number of DASD unit
26	38	IOBATAPE	DS X'0000'	Address of an alternate tape unit
28	40	IOBFLAG	DS X'0'	IOB flag
29	41		DS 3X'0'	Reserved for IBM use
2C	44	IOBWHATQ	DS F	Address of the double buffering queue that this IOB will service
30	48	IOBTHR	DS F	Address of I/O area being used by this job
34	52	IOBIRA	DS F	Interruption return address for overlapped I/O
38	56	IOBOUTBF	DS F	Address of anchor where THR will be enqueued
		IOBSIZE	EQU *-IOB	Address of an alternate tape unit

Figure 38. IOB (Input/Output Block) Format

Diagnostic Aids

Following is a list of the messages issued by the DASD Dump Restore Program. The associated label and method of operation figure are included in the list.

Message Code	Label	Figure	Message Text
DMKDDR536I	DDR536		rdev REPORTS DISABLED INTERFACE; FAULT CODE = code; NOTIFY CE
DMKDDR700E	DDR700		INPUT UNIT IS NOT A CPVOL
DMKDDR701E	DDR701	23	INVALID OPERAND - operand
DMKDDR702E	DDR702		CONTROL STATEMENT SEQUENCE ERROR
DMKDDR703E	DDR703	23	OPERAND MISSING
DMKDDR704E	DDR704		DEVICE rdev NOT OPERATIONAL
DMKDDR705B	DDR705		I/O ERROR rdev, CSW = csw, SENSE = sense, INPUT = {bbcchh|nnnnnn}, OUTPUT = {bbcchh|nnnnnn}, CCW = ccw
DMKDDR707E	DDR707		MACHINE CHECK
DMKDDR708E	DDR708		INVALID INPUT OR OUTPUT DEFINITION
DMKDDR709E	DDR709		WRONG INPUT TAPE MOUNTED
DMKDDR710A	DDR710		DEV rdev INTERVENTION REQUIRED
DMKDDR711R	DDR711		VOLID READ IS volid2 [NOT volid1]. DO YOU WISH TO CONTINUE? RESPOND YES, NO, OR REREAD:
DMKDDR712E	DDR712		NUMBER OF EXTENTS EXCEEDS 20
DMKDDR713E	DDR713		OVERLAPPING OR INVALID EXTENTS
DMKDDR714E	DDR714		RECORD {abbcchh/nnnnnn} NOT FOUND ON INPUT TAPE
DMKDDR715E	DDR715		LOCATION bbcchh IS A FLAGGED TRACK
DMKDDR716R	DDR716		NO VOL1 LABEL FOUND [FOR volid]. DO YOU WISH TO CONTINUE? RESPOND YES, NO OR REREAD:
DMKDDR717R	DDR717	25	DATA DUMPED FROM volid1 TO BE RESTORED TO volid2. DO YOU WISH TO CONTINUE? RESPOND YES, NO OR REREAD:
DMKDDR718E	DDR718		OUTPUT UNIT IS FILE PROTECTED
DMKDDR719E	DDR719		INVALID FILENAME OR FILE NOT FOUND
DMKDDR720E	DDR720		ERROR IN routine
DMKDDR721E	DDR721		RECORD {cchhr/nnnnnn} NOT FOUND
DMKDDR722E	DDR722		OUTPUT UNIT NOT PROPERLY FORMATTED FOR THE CP NUCLEUS
DMKDDR723E	DDR723		NO VALID CP NUCLEUS ON THE INPUT UNIT
DMKDDR724E	DDR724		INPUT TAPE CONTAINS A CP NUCLEUS DUMP
DMKDDR725R	DDR725	23, 25	DASD INPUT DEVICE WAS (IS) LARGER THAN OUTPUT DEVICE. DO YOU WISH TO CONTINUE? RESPOND YES OR NO:
DMKDDR726E	DDR726		MOVING DATA INTO THE ALTERNATE TRACK CYLINDER(S) IS PROHIBITED

Message Code	Label	Figure	Message Text
DMKDDR727E	DDR727		FLAGGED TRK track HAS NO PROPER ALTERNATE; SKIPPING THIS TRK
DMKDDR728E	DDR728	26	DECODE ERROR ENCOUNTERED: nn
DMKDDR729I			FULL TRACK READ FEATURE NOT AVAILABLE
DMKDDR731I	DDR731	28	COMPACT OPTION IGNORED FOR COPY OPERATIONS
DMKDDR756E	DDR706		PROGRAM CHECK PSW = psw
	MSG002	22	VM/370 DASD DUMP/RESTORE PROGRAM RELEASE n
	NEWADD		
	MSG02A		ENTER CARD READER ADDRESS OR CONTROL STATEMENTS
	MSG003		ENTER CYLINDER EXTENTS OR ENTER BLOCK EXTENTS
	MSG03B		ENTER NEXT EXTENT OR NULL LINE
	MSG005		END OF VOLUME CYL xxx HD xxx, MOUNT NEXT TAPE OR END OF VOLUME BLOCK xxxxxxxx, MOUNT NEXT TAPE
	MSG004		RESTORING xxxxxx
	MSG004		COPYING xxxxxx
	MSG004		DUMPING xxxxxx
	MSG004	27	PRINTING xxxxxx
	MSG001	24	END OF DUMP
	CLOSEJOB	24	END OF RESTORE
		24	END OF COPY
	MSG001	24	END OF PRINT
	EOJ		
	MSG001		END OF JOB
	RESPMSG		DO YOU WISH TO CONTINUE? RESPOND YES NO OR REREAD:
	RESPMSG2		DO YOU WISH TO CONTINUE? RESPOND YES OR NO

Figure 39. The DASD Dump Restore Program Messages

Index

Chapter 4. Installation Verification Procedure

Installation Verification Procedure

Introduction

The Installation Verification Procedure (IVP) for VM/SP is designed to exercise the generated system to verify that basic VM/SP facilities are operable. The IVP is contained in two files using the EXEC facility of CMS, and uses two virtual machines in addition to the system operator's virtual machine.

The tests exercise the following areas of CP:

- Multiple virtual machine support
- I/O spooling
- Transferring of spooled data to other virtual machines
- Offline I/O operations
- Sending of messages to the system operator
- Paging operations
- Task dispatching and scheduling
- Disk I/O support
- Automatic warm start following abnormal termination of VM/SP.

The following facilities of CMS are exercised:

- Normal CMS command processing
- Disk formatting
- Copying of files
- Creation and modification of files via EDIT command
- Assembly of executable programs
- Execution of user programs
- Creation and execution of user-written commands
- Printing and punching of CMS files
- Issuing of commands to CP
- Use of multilevel nested EXEC procedures
- Stacking and unstacking of command and data input from the terminal
- Communication with user from EXEC procedures.

Several other system facilities, incidental to the primary IVP tests, are exercised. Certain system facilities, VSAM and Access Method Services under CMS, are not exercised by the IVP.

The IVP requires operator intervention only when an operational decision is to be made, or to initiate the IVP tests themselves. All file creation, erasure, management, and logoff of the virtual machines (with the exception of the system operator) at test completion is performed automatically without operator or user action.

Method of Operation

This section describes the execution of the two EXEC procedures of the IVP (Installation Verification Procedure).

Figure 40 shows the relationship of the diagrams.

Figure 41 describes the highest level EXEC procedure, IVP.

Figure 42 describes the major functions of the nested EXEC procedure IVPX.

Figure 43 describes test procedure 1.

Figure 44 describes test procedure 2.

Figure 45 describes the error processing.

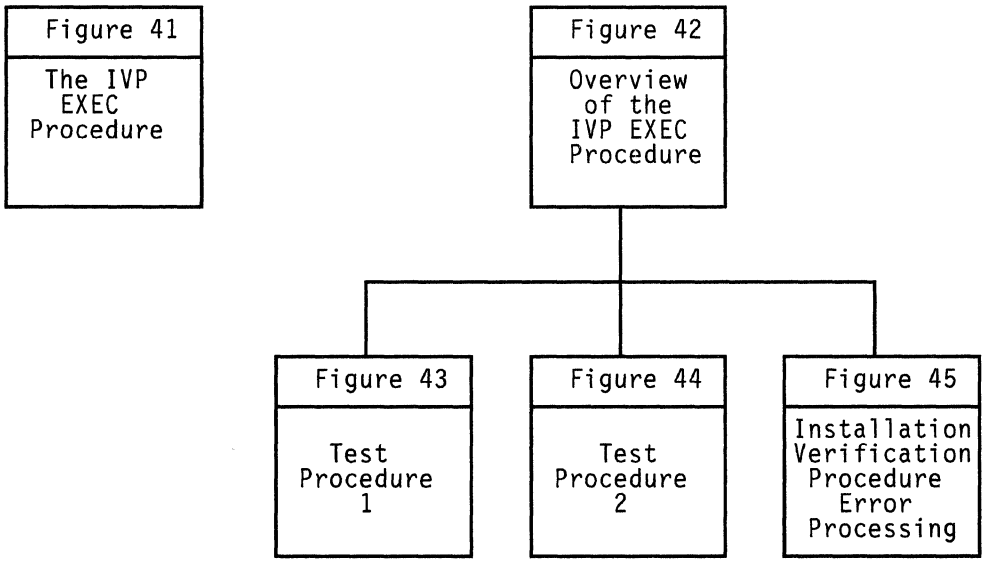


Figure 40. Key to the Installation Verification Procedure Method of Operation Figures

Installation Verification Procedure

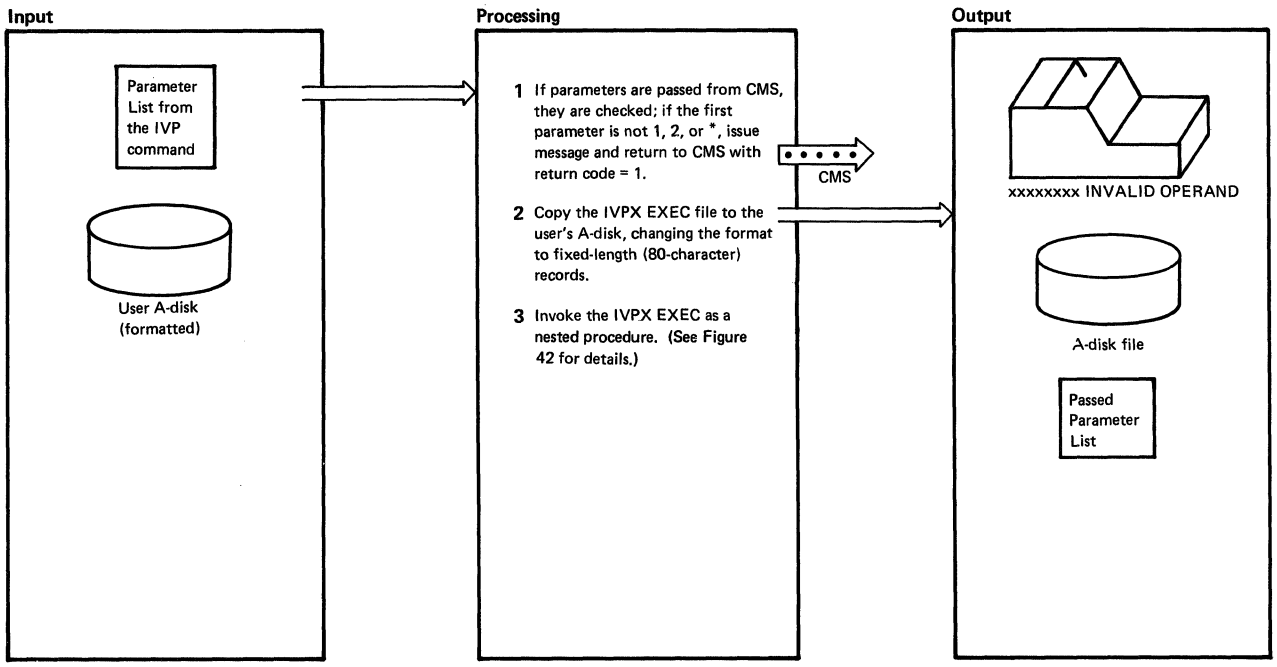
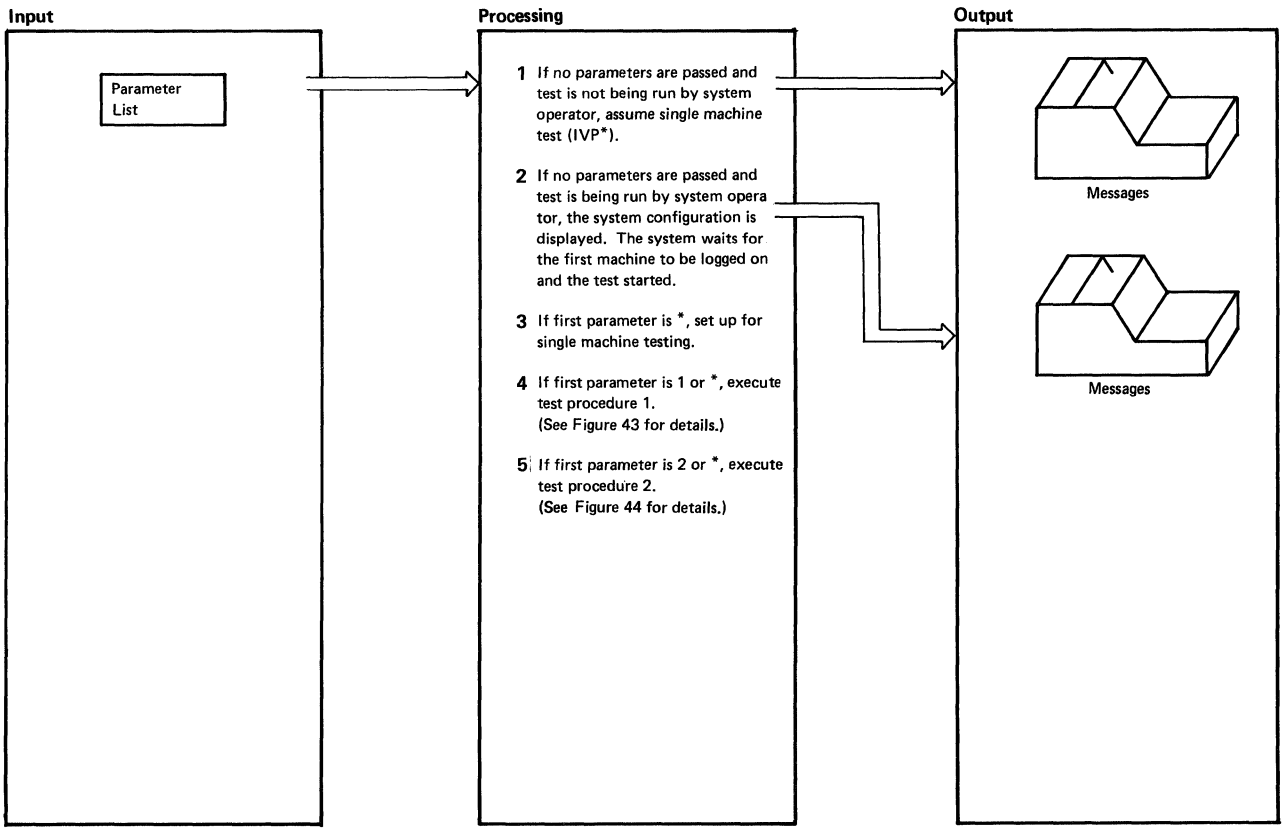


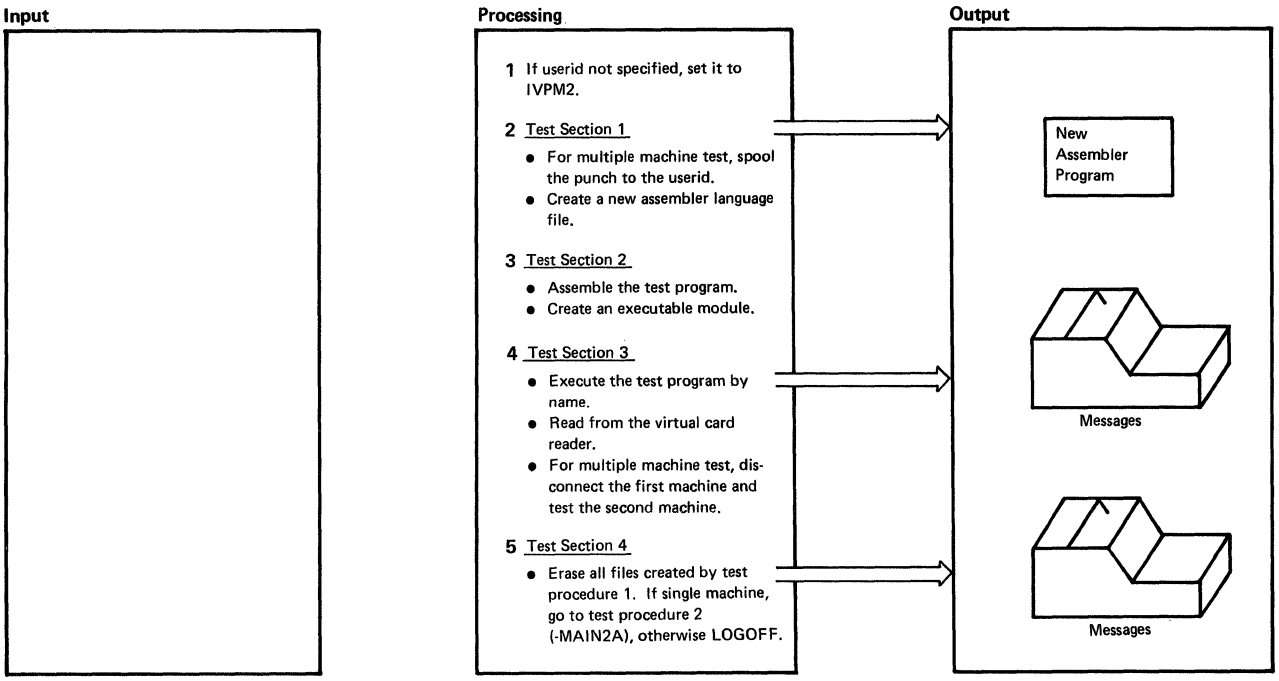
Figure 41. IVP EXEC Procedure



Notes		Module	Label	Ref
1	When no parameters are specified on the IVP command, the message ***ARE YOU THE SYSTEM OPERATOR? ENTER "YES" or "NO" is displayed. If the response is NO, the message ***NOT SYSTEM OPERATOR-DEFAULT TO IVP* is displayed. Single machine testing is set up (-INIT), and the testing starts at test procedure 1.	IVPX	CKOP	
2	The real system configuration is displayed. Then the virtual machine enters a dormant state which can be interrupted by signaling attention from the terminal. The message ***THIS PORTION OF IVP NOW GOING TO SLEEP is displayed and the system waits.	IVPX	CKOP	
3	Set &GLOBAL2=4 to indicate single machine test. Erase all CMS files with filenames IVPST and IVPTST2. If return code is other than 0 or 2, the ERASE command (to erase the EXEC file) is stacked in the terminal and control returns to the CMS command environment. If the return code is 0 or 2, test procedure 1 (MAIN1A) is executed.		INITB GETOUT	

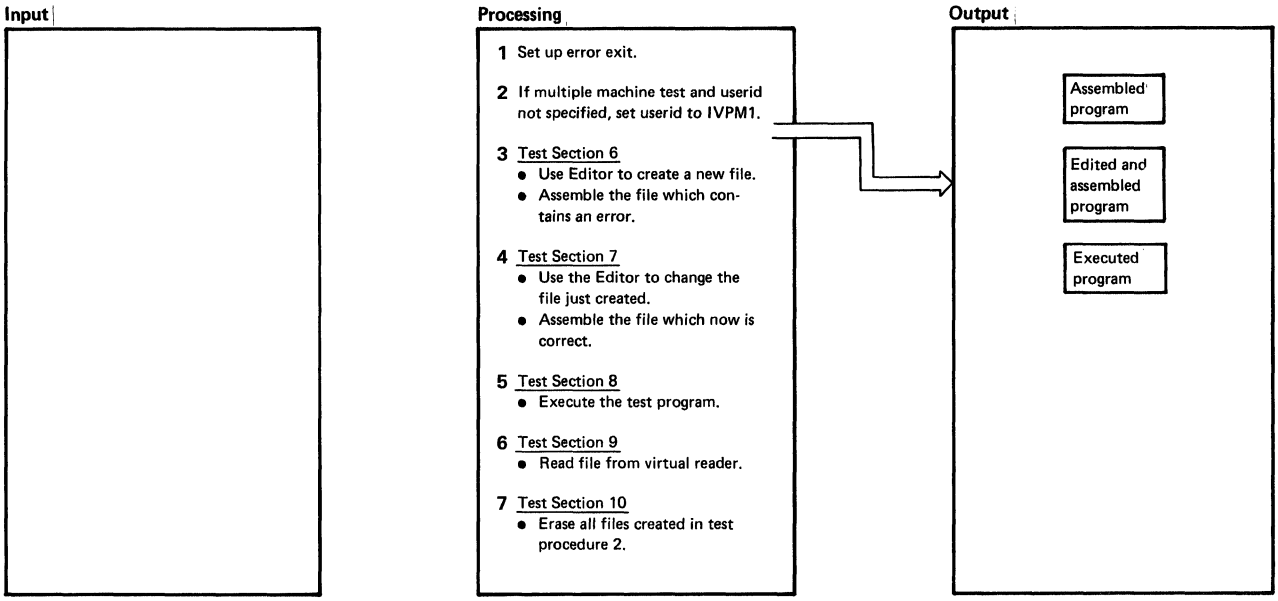
Figure 42. Overview of the IVPX EXEC Procedure

Installation Verification Procedure



Notes		Module	Label	Ref
1	For a multiple machine test, the userid is set to IVPM2 or to the userid specified as the second operand of the IVP command. When the userid is set to IVPM2, &GLOBALS is set to 2 to indicate the standard test.	IVPX	MAIN1	
2	The assembler language statements are stacked in the terminal input buffer and edited.	IVPX	MAIN1A	
3	The test program created in test section 1 is first assembled (ASSEMBLE command) and then made executable by issuing the LOAD and GENMOD commands.	IVPX	K256	
4	The test program, IVPTST, is executed. Next a READ is issued to the virtual reader and a return code is requested.	IVPX	LOOPA	
	If the return code is other than 0 or 8, the ERASE command to erase the EXEC file is stacked in the terminal, and control returns to the CMS command environment.		GETOUT	
	When testing multiple machines, the following message is issued: ***THIS PORTION OF IVP NOW GOING TO SLEEP			
	The first machine is then disconnected. The operator enters the above commands to start the second machine. The procedure loops (control keeps returning to -LOOPA) until the file to start the second machine is spooled to the reader. The STATE command is issued to verify the existence of the file. The second machine is started.		FINIS	
5	All the IVPTST files are erased. If the test machine is still connected (&GLOBAL2=3) the following messages are issued: ***TEST SECTION 5 RESERVED FOR FUTURE USE*** ***IVP TEST 1 SUCCESSFULLY COMPLETED These same messages are sent to the punch if the test machine is already disconnected (&GLOBAL=3).	IVPX	INLINE	
	The single machine test resumes at -MAIN2A, test procedure 2. If the standard test is running the message. ***IVP TEST 1 FINISHED is sent to the system operator. If &GLOBAL5=1, the test is running in 256K bytes of storage. If running machine tests, go to the LOGOUT routine. The following commands are stacked. ERASE IVPX EXEC A1 CP LOGOUT The LOGOUT routine closes all files including the punch containing the messages issued after test machine 1 was disconnected. The multiple machine test resumes at -MAIN2, test procedure 2.		LOGOUT	

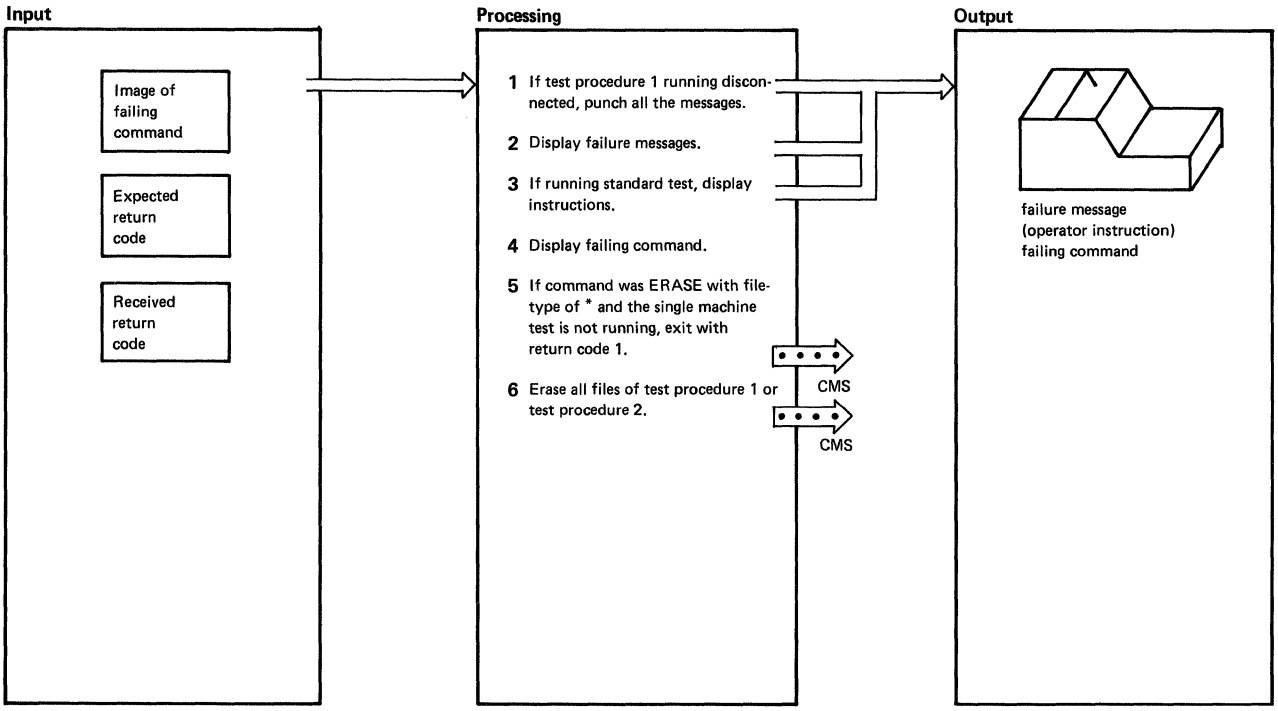
Figure 43. Test Procedure 1



Notes		Module	Label	Ref
1	Set the error exit to -FAIL2. For a single machine test, edit directly to the CMS command environment. The ERASE and LOGOUT commands are stacked in the terminal and the EXEC procedure exits with a return code of 1. Execution is now ended within the nested EXEC. The return code of 1 forces the next level EXEC to exit to the CMS command environment.	IVPX	-FAIL2	
2	For a multiple machine test, the userid is set to IVPM1 or to the userid specified as the second operand of the IVP command. When the userid is set to IVPM1, &GLOBAL5 is set to 2 to indicate the standard test.		MAIN2	
3	The input data is stacked for the editor, which creates the IVPST2 ASSEMBLE file. The file just created is assembled. Error 8 occurs because the ASSEMBLE file contains one error.	IVPX	MAIN2A	
4	The statement in error is corrected. The file is then assembled. Since the error is corrected the TEXT file is created.	IVPX	MAIN2A	
5	The test program is loaded and then started.	IVPX	LOOP LOOP2	
6	The file is read from the virtual reader. If there is no file in the reader on the first loop, a file is created, punched, and spooled to the reader. For a single machine test, a dummy message file is created, punched, and spooled to the reader on the same machine. For a multiple machine test, the messages are spooled to the reader on the userid system. The input is stacked in the terminal fof the editor. A dummy message is edited and punched. Control returns to -LOOP. The STATE command is issued to be sure the file is successfully read onto disk. The contents of the file are displayed. For multiple machine standard test, the message DON'T START SPOOL DEVICES UNTIL TOLD is sent to the system operator. The multiple machine test determines that the file was successfully read and punches and prints that file.	IVPX IVPX	LOOP1 NOSPL	
7	All files are erased and messages are displayed. ***IVP TEST 2 SUCCESSFULLY COMPLETED ***IVP PROCEDURE FINISHED If a single machine test, the command to erase the EXEC file is stacked in the terminal and control returns to the CMS command environment. If a multiple machine test, the commands to erase the EXEC file and LOGOUT are stacked for CMS. If running the standard test, the messages ***IVP TEST NOW FINISHED ***SIGNAL ATTN AND ENTER: BEGIN are sent to the system operator. For the multiple machine test, control then returns to the CMS command environment.	IVPX	GETOUT	

Figure 44. Test Procedure 2

Installation Verification Procedure



Notes		Module	Label	Ref
1	If test machine 1 is disconnected, the messages are sent to the punch, rather than the virtual machine console.	IVPX	CHECK1	
2	The message ***IVP FAILURE HAS OCCURRED*** is displayed.	IVPX		
3	The messages IVP HAS FAILED - REPLY NO TO ABORT MESSAGE ***SIGNAL ATTN AND ENTER: BEGIN are sent to the system operator.	IVPX	CHECK2	
4	The messages ***COMMAND:xxxxxxxx ***EXPECTED RETURN CODE xxx ***RECEIVED RETURN CODE xxx are displayed.	IVPX		
5	Control returns to the next level EXEC procedure and the return code of 1 forces that level to return to the CMS command environment.	IVPX	QUIT	
6	If the number of the test section is less than 6, all the IVPTST files are erased. If the number of the test section is greater than 5, all the IVPTST2 files are erased. Because this is a nested EXEC procedure, exit with a nonzero return code. A nonzero return code forces the next level EXEC to return to the CMS command environment.			

Figure 45. Installation Verification Procedure Error Processing

Program Organization

The IVP (Installation Verification Procedure) consists of two EXEC procedures: IVP and IVPX. Figure 46 shows the structuring of the major routines of the IVP. Figure 47 on page 104 and Figure 48 on page 104 relate the test sections to the CP or CMS functions being exercised.

Installation Verification Procedure Routine Structuring

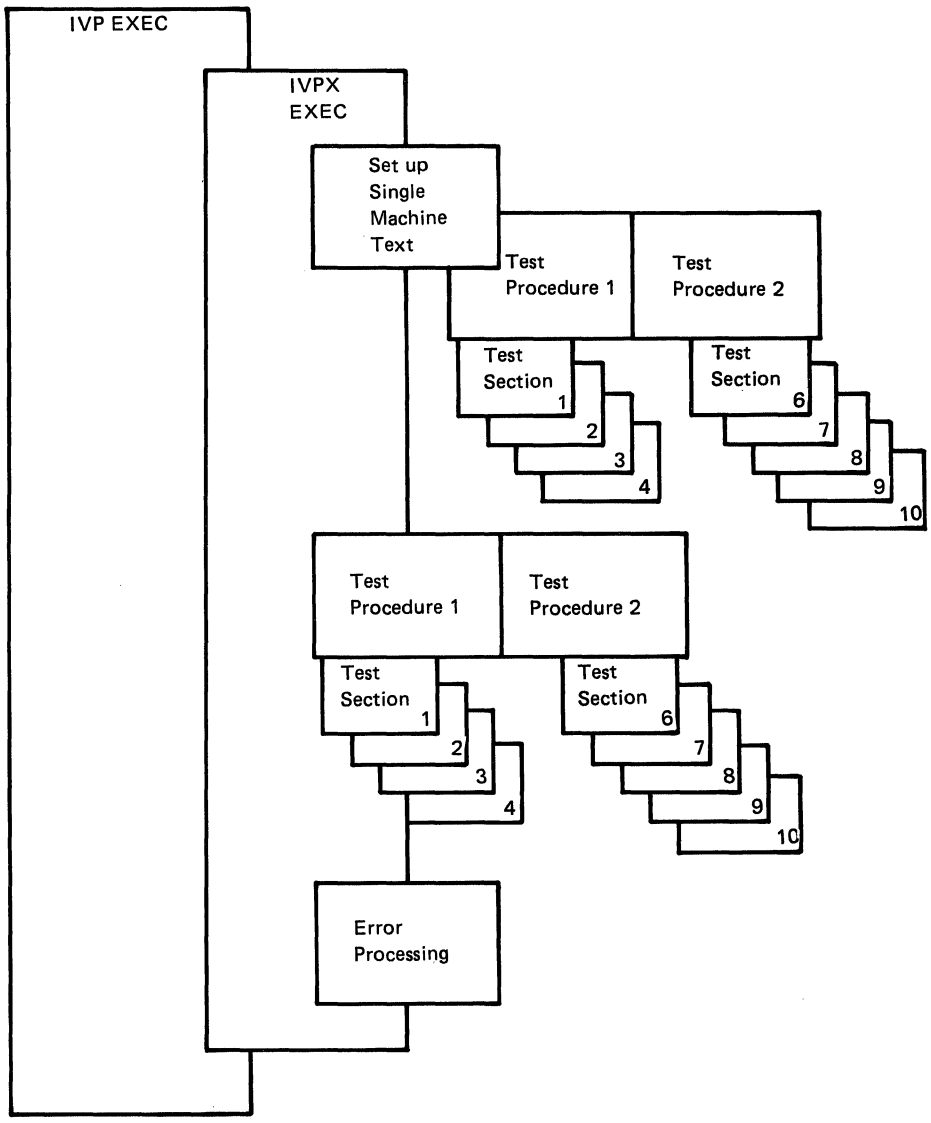


Figure 46. Structure of Installation Verification Procedure Routines

Installation Verification Procedure Testing (CP)

Function Tested	Test Section and Comments
Multiple virtual machine support	Test Procedures 1 and 2 test multiple virtual machine support when IVP * is not specified or assumed.
I/O Spooling	Test Section 9.
Transferring of spooled data to other virtual machines	Test Section 9 when IVP * is not specified or assumed
Offline operations	Test Section 9.
Sending messages to system operator	Test Sections 4 and 9.
Page operations	Used throughout IVP.
Task dispatching and scheduling	Used throughout IVP.
Disk I/O support	Used throughout IVP.
Automatic warm start	Error processing.

Figure 47. Installation Verification Procedure Testing for CP

Installation Verification Procedure Testing (CMS)

Function Tested	Test Section and Comments
Command processing	Used throughout IVP
Copying of files	The IVP EXEC procedure
Creation and modification of files via EDIT command	Test Sections 1, 6, and 7
Assembly of executable modules	Test Sections 2, 6, and 7
Execution of user programs	Test Sections 3 and 8
Creation and execution of user-written commands	Test Section 3
Printing and punching of CMS files.	Test Section 7
Multilevel EXEC procedures	Used throughout IVP

Figure 48. Installation Verification Procedure Testing for CMS

Directory

Following is an alphabetical list of the labels in the IVPX EXEC procedure. The associated method of operation diagram and a brief description of the function performed at the point in the program indicated by each label are included in the list.

Label	Figure	Description
CHECK1	45	Sends messages to punch when machine is disconnected
CHECK2	45	Displays the failing command
CKOP	42	Sets up for execution when IVP is invoked without any parameters specified
FAIL2	44	Exits to CMS command environment if single machine test is running
FINIS	43	End of Test Procedure 1.
GETOUT	42, 43, 44	Error exit for single machine test.
INITB	42	Sets up for single machine test
INLINE	43	Erases all files created during Test Procedure 1
K256	43	Assembles and executes the program created in Test Section 1
LOGOUT	43, 44	Error exit for multiple machine test
LOOP	44	Reads file from the virtual reader during Test Procedure 2
LOOPA	43	Reads from the virtual reader during Test Procedure 1
LOOP1	44	Checks that file is read to disk successfully
LOOP2	44	Creates file, punches it, and spools it to reader when there is no file in the reader
MAIN1	43	Beginning of Test Procedure 1
MAIN1A	43	Point in Test Procedure 1 where the single machine test begins
MAIN2	44	Beginning of Test Procedure 2
MAIN2A	44	Point in Test Procedure 2 where the single machine test begins
NOSPL	44	Erases all files created in Test Procedure 2
QUIT	45	Abnormal end exit from a nested EXEC procedure

Diagnostic Aids

Following is a list of all the messages that the IVPX EXEC procedure issues, the label nearest to the point where the message is issued, and the associated method of operation diagram.

Label	Figure	Message Text
CKOP	42	*** ARE YOU THE SYSTEM OPERATOR? ENTER "YES" OR "NO."
CKOP	42	*** NOT SYSTEM OPERATOR - DEFAULT TO IVP *
CKOP	42	*** THIS PORTION OF IVP NOW GOING TO SLEEP.
		*** STARTING SYSTEM ABORT ROUTINE.
ABMSG		*** ENTER "GO" TO CONTINUE OR "NO" TO QUIT.
ABMSG		*** THIS IS THE LAST STEP OF THE IVP PROCEDURE.
		*** FOLLOWING SYSTEM RESTART (WARM START), START SPOOLING DEVICES
ABMSG		MANUALLY DEPRESS CPU RESTART KEY TO ABORT SYSTEM.
PERFORM		*** STARTING TEST SECTION x
CHECK1	45	*** IVP FAILURE HAS OCCURRED ***
CHECK1	45	*** IVP HAS FAILED - REPLY NO TO ABORT MESSAGE
		*** SIGNAL ATTN AND ENTER: BEGIN
CHECK2	45	*** COMMAND: xxxxxxxx
		*** EXPECTED RETURN CODE xxx
		*** RECEIVED RETURN CODE xxx
INLINE	43	*** TEST SECTION 5 RESERVED FOR FUTURE USE ***
INLINE1	43	*** IVP TEST 1 SUCCESSFULLY COMPLETED
		*** IVP TEST 1 FINISHED
LOOP1	44	DON'T START SPOOL DEVICES UNTIL TOLD.
NOSPL	44	*** IVP TEST 2 SUCCESSFULLY COMPLETED
		*** IVP PROCEDURE FINISHED
NOSPL	44	*** IVP TEST 2 FINISHED
		*** SIGNAL ATTN AND ENTER: BEGIN

Index

Installation Verification Procedure

Chapter 5. Generating and Updating VM/SP

Introduction

The VM/SP update facility provides for the updating of files with several levels of updates and any number of program temporary fixes (PTFs). For Assembler language source statement files, procedures are supplied for assembling the updated source code to produce a uniquely defined text deck. The deck has a unique name and some control cards to identify the origin of the updates, macro libraries, and source statements. For macro library files, a copy file is produced to identify the origin of the input and any updates applied.

Procedures are provided for generating load files from various object modules, for generating MACLIB files from various COPY and MACRO files, and creating text libraries from TEXT files.

The procedure for updating VM/SP has a file naming convention for update and text files, a set of programs to support the processing, and a set of EXEC procedures and modules to process the files.

- The VMFASM procedure incorporates PTFs or updates.
- The VMFLOAD procedure creates a new CP or CMS nucleus.
- The VMFMAC procedure generates a new macro library.
- The VMFNLS procedure updates national language related files.
- The VMFTXT procedure creates a new text library.

Update Files

Files used to update another file are given a filetype of UPDTxxxx, where xxxx is a unique update identifier for programmer and system use. The filename of the update file must be the same name as the file to be updated. For instance, the file PROGRAM ASSEMBLE could be updated by the file PROGRAM UPDTGN30 or the file PROGRAM UPDTGC61.

The creation and use of update files are described in the UPDATE command discussion in the *VM/SP CMS Command Reference.*

TXT Files

Text files are produced by the assembler as a part of the VMFASM procedure. The filename of the text file is the same as the filename of the ASSEMBLE file. The filetype of the completed text deck is TXTnamex, where 'namex' represents a unique update level identifier. The value of 'namex' is taken from a control file, and corresponds to the highest level of update applied. In addition, the text deck is produced from a combination

of the assembler text deck and an auxiliary control file containing data describing the origin of the files used. The auxiliary file is called 'filename UPDATES' and is produced by a program called VMFDATE. The filename is the same as the filename of the UPDTxxxx file.

Control Files

Each user may have several control files to specify various combinations of updates and macro libraries to be used. A control file must have a filetype of CNTRL. These control files contain records in the following format:

```
nam00 MACS maclib1 maclib2 ...
nam01 UPDTup1
nam02 UPDTup2
nam03 UPDTup3
nam04 AUXxxxxx
```

The suffixes up1, up2, up3, and xxxxx are update identifier fields, and the fields nam00, nam01, nam02, nam03, and nam04 are update level identifiers.

The first record is the MACS record that lists in search order the macro libraries (maclib1 maclib2...) to be used in the assembly. Up to 29 libraries may be specified (subject to the character limit of the MACS record line).

Records 2, 3, and 4 are update identification records. They define the UPDTxxxx files that were created (via update control cards and source statements) to update some particular file. Record 2 defines an UPDTup1 file, and records 3 and 4 define UPDTup2 and UPDTup3 updates, respectively. None, some, or all of the updates may exist to be applied.

Record 5 defines an auxiliary file that specifies an auxiliary list of PTFs or updates that are to be applied. Record 5 defines an auxiliary file identified as 'filename AUXxxxxx', where 'filename' is the same as the filename of the input file and xxxxx is an update identifier (the update identifier for an auxiliary control file cannot be "aux"). Records in the auxiliary file have the following format for PTFs to be applied:

```
PTF  PTF001   comments
   PTF002
PTF  PTF003
*    Any comment
```

The PTF field is an optional identifier, and the second field (for example, PTF001) defines a specific PTF to be applied. The PTF has a 'filename PTF001' identification, where 'filename' is the same as the filename of the file to be updated. The comment field is used to describe the function of the particular PTF. The * record is ignored and is used to provide additional comments on any updates or PTFs.

The updates (PTFs included) are applied in the reverse order in which they appear. In the previous example, the updates would be applied in the following order:

```
PTF003
PTF002
PTF001
UPDTup3
UPDTup2
UPDTup1
```

The PTF records can be directly included in the CNTRL file if desired, but it is usually more convenient to place them in a separate auxiliary (AUXxxxxx) file.

There can be any number of UPDTxxxx definition and auxiliary control file definition records, but only one MACS record. The complete CNTRL file can have any filename, but typically has the same name as the first specified UPDTxxxx control record. In the example, the file could be named UP1 CNTRL.

The underlined fields in each record mark the level identification fields. The highest level (last) update to be applied selects the name that can be used to identify updated files. In the example, if UPDTup3 was the last update applied, then the name selected would be nam03. The value for the identification usually consists of a combination of the update identifier up1, up2, ... (up to four characters) and additional characters up to a maximum of 5 for the combined update identifier and additional characters. If no updates are applied, then the nam00 field is selected to identify the TXTnam00 produced. This name can be used to uniquely identify updated files. The text files described above, for instance, can have a filetype of TXTup3. It is desirable, on occasion, to have entries in the user CNTRL file that specify a level identification but no update. A record of the following format, for example, is allowed:

```
nam05
```

This is because the control file serves a double purpose and is used for loading text decks as well as updating input files. An identifier of TEXT as a name causes special handling in the VMFASM EXEC procedure, whether or not an update is used with it. A name of TEXT is used without level identification catenation. Thus, TEXT becomes the filetype.

System EXEC Procedures

Several system control files provide for system update and creation. Some EXEC procedures invoke others or make use of user-supplied control files to accomplish various functions such as multilevel updating, text generation, and macro library generation.

VMFASM EXEC Procedure

The VMFASM EXEC procedure performs the multilevel update function by invoking the DMSUPD module (via the CMS UPDATE command) before assembling the desired files. To update and assemble a source file, the VMFASM EXEC is invoked as a CMS command as follows:

```
VMFASM fn ctlfile [options]
```

where:

`fn` is the filename of the source file to be updated. It must have a filetype of ASSEMBLE.

`ctlfile` is the filename of the control file. The control file must have a filetype of CNTRL. This control file contains the MACs (macro library), update, and any AUXxxxx control records.

The VMFASM procedure invokes the DMSUPD module via the CMS UPDATE command, passing the values 'filename', 'ASSEMBLE', and 'control'.

The UPDATE command returns a level identifier and a MACLIB list from the MACS record of the control file. If the identifier is TEXT, then that becomes the filetype of the complete text deck; otherwise the filetype is TXTxxxxx (for example, TXTup3m1). The EXEC procedure then reads the MACLIB list passed by UPDATE and issues a GLOBAL command to prepare for the assembly using the specified libraries.

Options VMFASM accepts only the nondefaulted options. All other assembler options entered are ignored and the defaults are used.

The defaults are: PRINT, NOTERM, LIST, NODECK, NORENT, SYSPARM(), and XREF(FULL). The options that can be specified for the VMFASM EXEC are: DISK, NOTERM, NOLIST, DECK, RENT, EXP, XREF, and RLD. The defaults for the VMFASM EXEC are: PRINT, TERM, LIST, NODECK, NORENT, SYSPARM(SUP), XREF(SHORT), and NORLD.

These options are described in the *VM/SP Installation Guide*.

The VMFDATE program is used to construct a record for each MACLIB used and for the ASSEMBLE file. Each record is placed in the auxiliary file 'filename UPDATES'. The text deck produced by the assembler is combined with the file produced by the VMFDATE program and is named 'filename TXTxxxxx', where 'filename' is that of the ASSEMBLE file, and 'TXTxxxxx' is constructed from the update level identifier returned by the UPDATE command. All intermediate files are erased, leaving only the original ASSEMBLE and UPDTxxxx files, and the newly created text file.

VMFLOAD Procedure

The VMFLOAD procedure uses two user-supplied files, a loadlist EXEC an an optional control file identical in format to the CNTRL file used by VMFASM and UPDATE, to produce a punched deck comprised of several text files. VMFLOAD is invoked as a CMS command as follows:

```
VMFLOAD loadlist ctlfile [langid]
```

where:

`loadlist` is the filename of an EXEC file that contains the names of object modules in the order in which they are to reside in the complete load file for the nucleus. For example:

```
&CONTROL OFF
&1 &2 fn [ft] [(LANG]
&1 &2 fn [ft] [(LANG]
 .
 .
 .
```

where `fn` and optionally `ft`, are the filename and filetype of an object module to be punched. The object modules are punched in the order specified, beginning at the top of the loadlist EXEC. If a filetype is specified, VMFLOAD searches for that specific file, and, if it finds it, punches it without a header card.

LANG is a special option you use for national language-related files, such as message repositories. Any entry with the LANG option is punched with a header card. If you specify `langid` on the VMFLOAD command, VMFLOAD determines the filetype of the object module file you want to punch.

If the filetype is not specified in the loadlist, VMFLOAD uses the control file to determine which object module is the highest level object module available. VMFLOAD searches the control file from top to bottom. When it finds the appropriate object module, VMFLOAD punches it.

`ctlfile` is the filename of the control file. This file lists update files or auxiliary control file to be updated. This is usually the same control file used to apply updates to modules via the VMFASM or UPDATE commands. This file identifies the highest level object module available, if the filetype is not specified in the loadlist.

`langid` is the identifier for national language-related files that are loaded into the nucleus. VMFLOAD constructs the filetype identifier TXTlangid and uses TXTlangid as the filetype for any modules marked with LANG in the loadlist.

VMFLOAD uses the control file to search for the desired text deck in the order in which the identifiers are specified in the file. The first file located

is punched, and all lower files are ignored. If the end is reached without finding a text file, VMFLOAD displays the message 'filename TEXT' NOT FOUND, and continues processing with the next entry in the loadlist EXEC. It is quite possible to have a completed load deck comprised of different levels of text decks.

DMKLD00E Service Program

The loader (DMKLD00E) is a service program that is used to generate a CP, CMS, or RSCS nucleus. The loader loads the text decks supplied with it, resolves CCW addresses, and resolves address constants. The same loader is used whether a virtual=real or standard CP system is generated.

The loader is distributed with the following default I/O addresses:

Console=009
Printer=00E

These addresses can be overridden by a control card that must be placed after the last loader card and the first card of the text decks. The format of the control card is:

Column	Contents
1	12-2-9 punch
2-4	DEV
5	blank
6-13	PRNT=xxx (xxx is the printer address)
14	blank
15-22	TYPW=xxx (xxx is the console address)

The format of the other control cards can be found in the discussion of the LOAD command in the *VM/SP CMS Command Reference*.

The loader is self-relocating, that is, it is initially loaded at address 8000 (decimal); it then relocates itself to the top of storage. (For example, if the size of the loader is 10K, and the storage size of the system is 256K, the loader will occupy the area of storage between 246K and 256K.) After relocating itself, the loader clears the storage it was originally loaded in. As the loader needs free storage to perform its operations, it extends downward through storage.

The text decks being loaded must not try to overlay either the loader or any address between zero and 100 (hexadecimal). The text decks are loaded into storage in a positive direction (that is, upward through storage). If the text decks are going to overlay the loader's free storage, the operation is terminated.

VMFMAC EXEC Procedure

The VMFMAC EXEC procedure applies updates to copy or macro files and builds a new macro library. The VMFMAC EXEC procedure is invoked as a CMS command as follows:

```
VMFMAC libname [ctlfile]
```

where:

`libname` is the filename of the macro library to be updated, and of the EXEC file that contains the names of the library members. The entries in libname EXEC must be in the following format:

```
&1 &2 fn1
&1 &2 fn2
 .
 .
 .
```

where fn1, fn2, and so on, are filenames of macro or copy files to be updated and included in the macro library, which must have a filetype of MACLIB.

`ctlfile` is the filename of an optional control file (if one exists) to be used to apply the updates. The filetype must be CNTRL. The filenames used by VM/SP are DMKSP, DMKPSA, DMKSPM, DMSSP, and DMSMSP.

The UPDATE command is issued for each macro or copy file. If the update procedure is successful, the member is added to the NEWMAC MACLIB. After all macro and copy files have been processed, any existing libname MACLIB file is erased and the NEWMAC MACLIB is renamed to libname MACLIB.

VMFNLS Procedure

The VMFNLS EXEC automatically applies updates to source files, generates text files, and renames them so they can be loaded into the system. The format of the VMFNLS command is:

```
VMFNLS fn  ft  [(options. ... [)]]
```

where:

`fn` is the filename of the source file that is to be converted to text.

`ft` is the filetype of the source file that is to be converted to text. Only REPOS, DLCS, and ASSEMBLE are allowed.

`cntrl` is the name of the control file that is used to apply updates to the source file before text is generated.

`options` are options for the three commands that VMFNLS can issue. These commands are GENMSG, CONVERT COMMANDS, and ASSEMBLE.

The VMFNLS exec does different tasks, depending on the type of input source file.

If the input source file is a message repository file or a command syntax definition file:

- VMFNLS applies updates to the source file, producing the file *$fn ft*. If necessary, VMFNLS changes the filename of this temporary *$fn ft* file to match the filename required for the text file; it does not use the filetype, however.

- VMFNLS then determines the langid associated with the source file. If the source filename is only six characters, VMFNLS assigns the langid AMENG as a default; otherwise, it extracts the country code from the 7th and 8th characters of the source filename.

  VMFNLS LANGLIST contains a list of all valid country codes, along with the associated langid and language name. VMFNLS uses this list to convert the source filename to the text filename.

- Next, VMFNLS compiles the source file with the appropriate command.

  1. If the source file is a message repository, it has a filetype of REPOS. VMFNLS invokes GENMSG to produce a text file and a listing file from the source file. The text file that has the same filename as the input file, and a filetype of TXT*langid*. The listing file has the same filename as the text file; however, VMFNLS changes it to instead match the filename of the source file.

  2. If the source file is a definition language for command syntax (DLCS) file, it has a filetype of DLCS. VMFNLS invokes the CONVERT COMMANDS command to produce two text files from this input file. The filenames of these text decks depend on the :DLCS statement contained with the input file. This statement identifies the applid, langid, and whether the input file is a user or system DLCS file.

     For a system DLCS file, the filenames of the text decks are *applid*SPA for the command syntax definition file and *applid*SSY for the translation and synonym table. For a user DLCS file, the filenames of the text decks are *applid*UPA for the command syntax definition file and *applid*USY for the translation and synonym table.

     CONVERT COMMANDS assigns the filetype TXT*langid* to the text files.

- VMFNLS appends the summary of updates to the front of the text file that is produced.

If the input source file is an ASSEMBLE file:

VMFNLS invokes the VMFASM EXEC to apply updates to the source, sends the update log to the printer, and produces an associated text deck with a filetype of TEXT. It also determines the langid associated with the source file.

VMFTXT EXEC Procedure

The VMFTXT EXEC procedure creates text libraries. VMFTXT rebuilds a named TXTLIB file using a member list in an EXEC file with the same name. The VMFTXT EXEC is invoked as a CMS command as follows:

```
VMFTXT libname [ctlfile]
```

where:

`libname` is the filename of the text library you want to update, and of the EXEC file that contains the names of the library members. The recommended format of the EXEC file is as follows:

```
&TRACE OFF
*Optional comments may be included
 &1 &2 [&3] fn [ft] [(FILename [) ] ]
 &1 &2 [&3] fn [ft] [(FILename [) ] ]
   .
   .
   .
```

where fn and optionally ft, are the filename and filetype of an object file you want to add to the library.

- If you specify a filetype, VMFTXT looks for the specific file.
- If you do not specify a filetype, and you do not specify a CNTRL file, then VMFTXT looks for a filetype of TEXT.
- If you do not specify a filetype, but you do specify a CNTRL filename, VMFTXT searches those filetypes for the specified member.

Each entry in the member list EXEC file may also specify an optional filename parameter (FILename) to be passed directly to the TXTLIB command. This parameter indicates that the member name is to be taken from the filename and not from the CSECT name within the file.

`ctlfile` is the filename of an optional file which VMFTXT uses to determine the filetypes of the object files being added to the text library. The filetype of the *ctlfile* must be CNTRL.

This is usually the same control file used to apply updates to modules using the VMFASM or UPDATE commands. This file identifies the filetype search order if you do not specify the filetype in the member list.

VMFTXT uses the EXEC file to determine which members to include. VMFTXT takes the files from the member list and adds them to the library. They are added in the order they appear in the member list.

If you specify a filetype in the member list, then VMFTXT adds that specific file. Normally you do not specify a filetype, in which case VMFTXT uses the update level identifier in the control file to determine the filetype.

For each member VMFTXT adds, a message is issued verifying this. When all the members have been added, a message is issued stating that member VMFTXT TEXT has been added.

Method of Operation

This section describes the following procedures for generating and updating VM/SP:

- Update procedure
- Nucleus loading facility
- The MACLIB generation facility.

Figure 49 shows the relationship of the diagrams.

Figure 50 shows the major functions of the VMFASM procedure.

Figure 51 shows the initialization of the VMFASM procedure.

Figure 52 describes the assembling portion of the VMFASM procedure.

Figure 53 describes the VMFDATE program.

Figure 54 describes the major functions of the DMSUPD (update) program.

Figure 55 describes the operand and option checking for the Update program.

Figure 56 describes the multiple level update procedure.

Figure 57 describes the processing of control records for the Update program.

Figure 58 describes the single level update procedure.

Figure 59 shows how inserting is done.

Figure 60 describes the exit procedure for the Update program.

Figure 61 describes the VMFLOAD (Nucleus load) program.

Figure 62 describes the procedure that builds the MACLIB.

Figure 63 describes the procedure for updating national language related files.

Figure 64 describes the procedure that builds the TXTLIB.

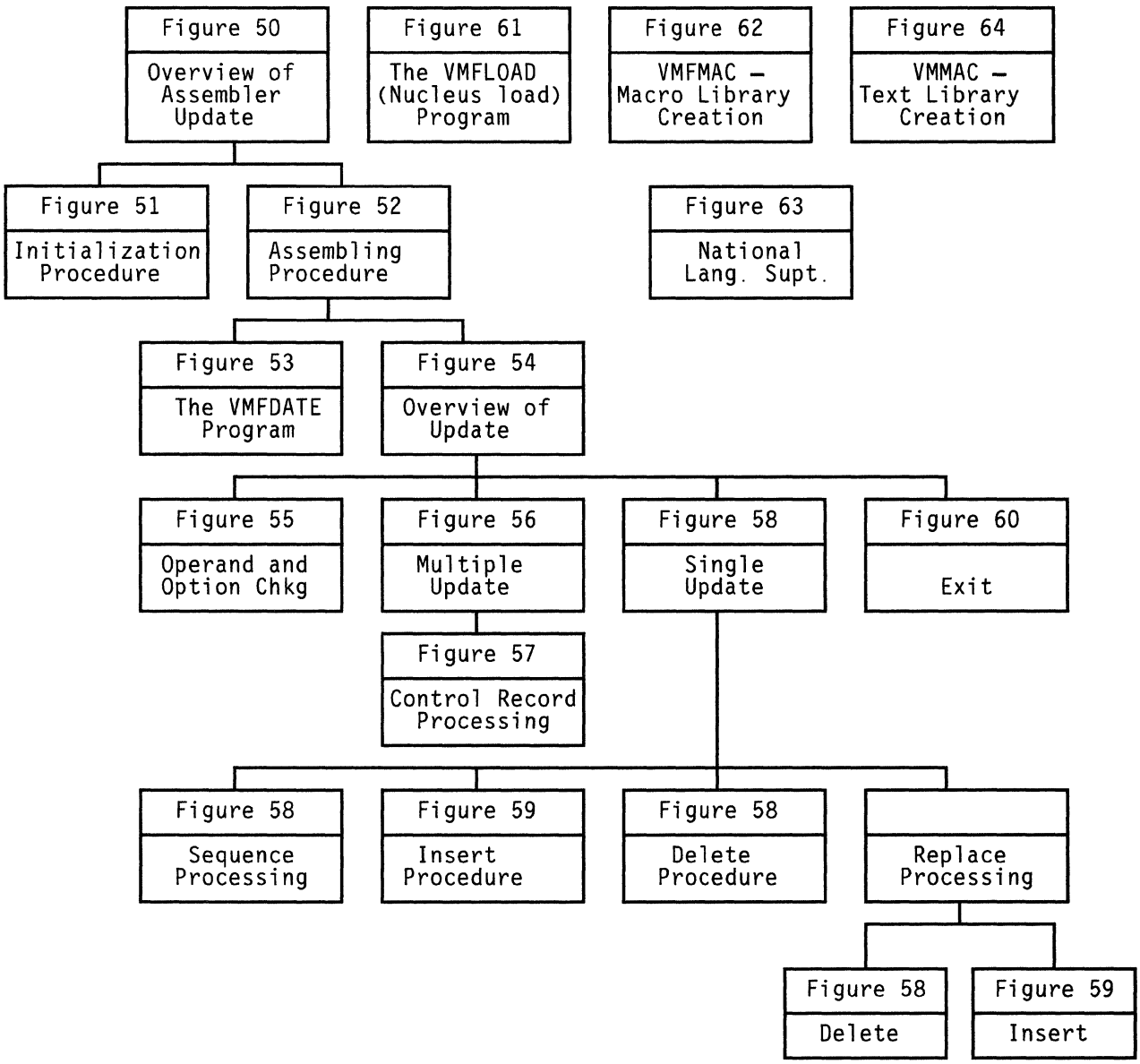


Figure 49. Key to the Procedures for Generating and Updating

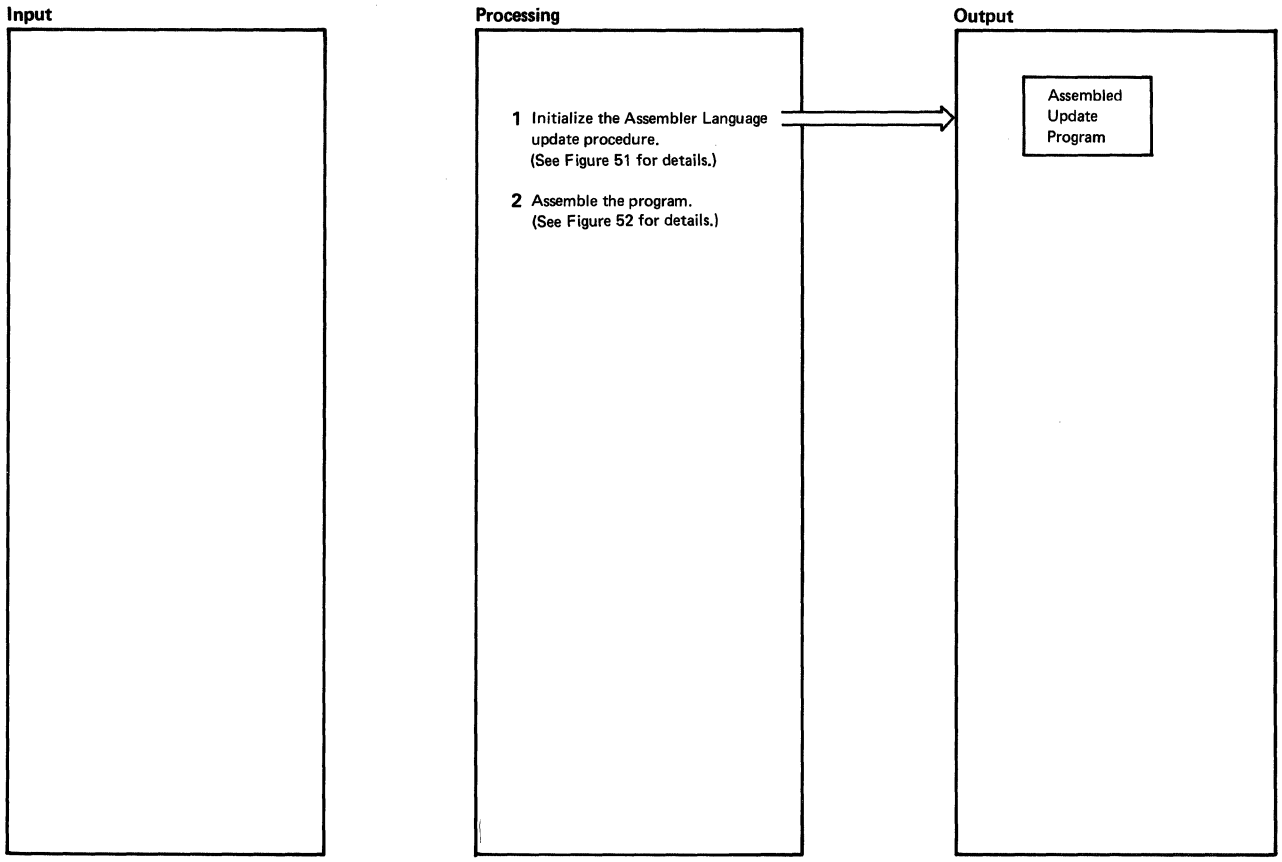
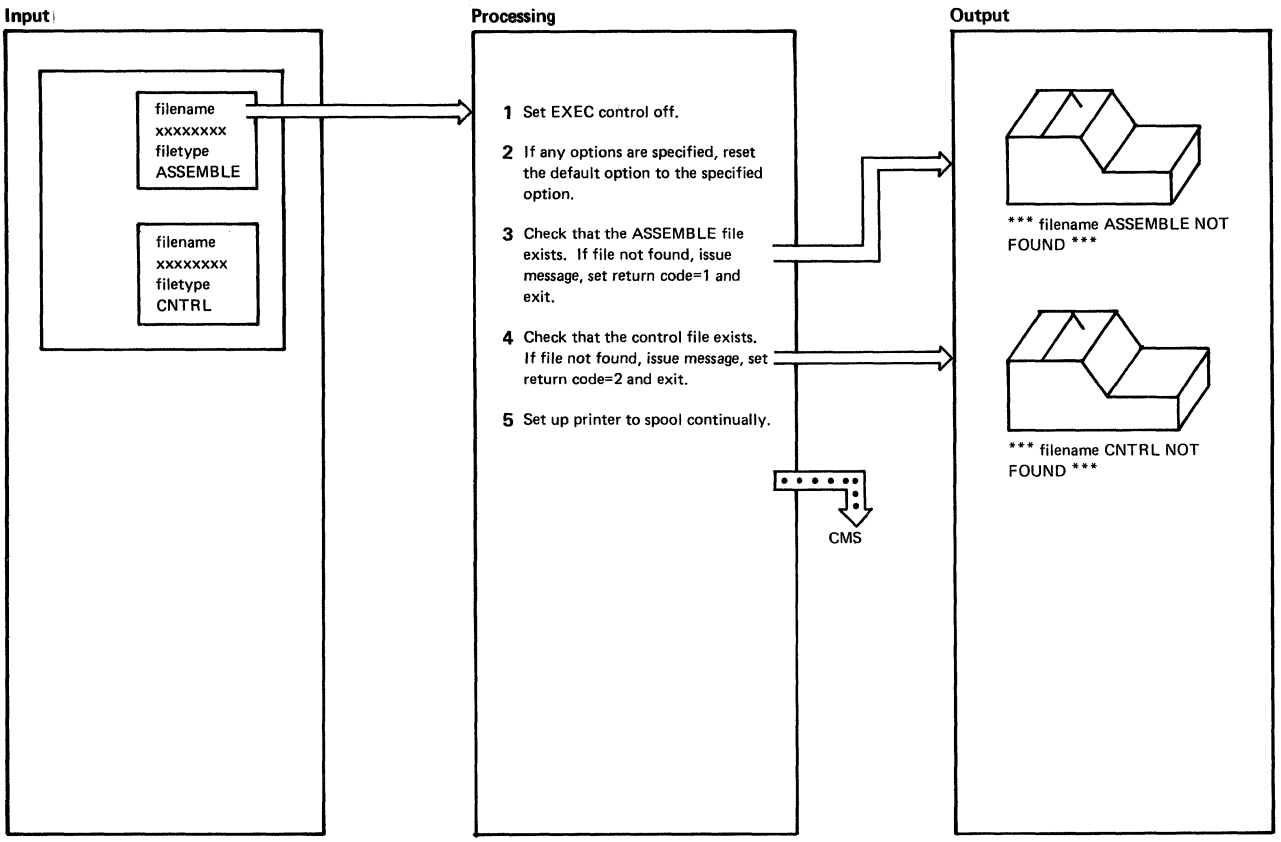
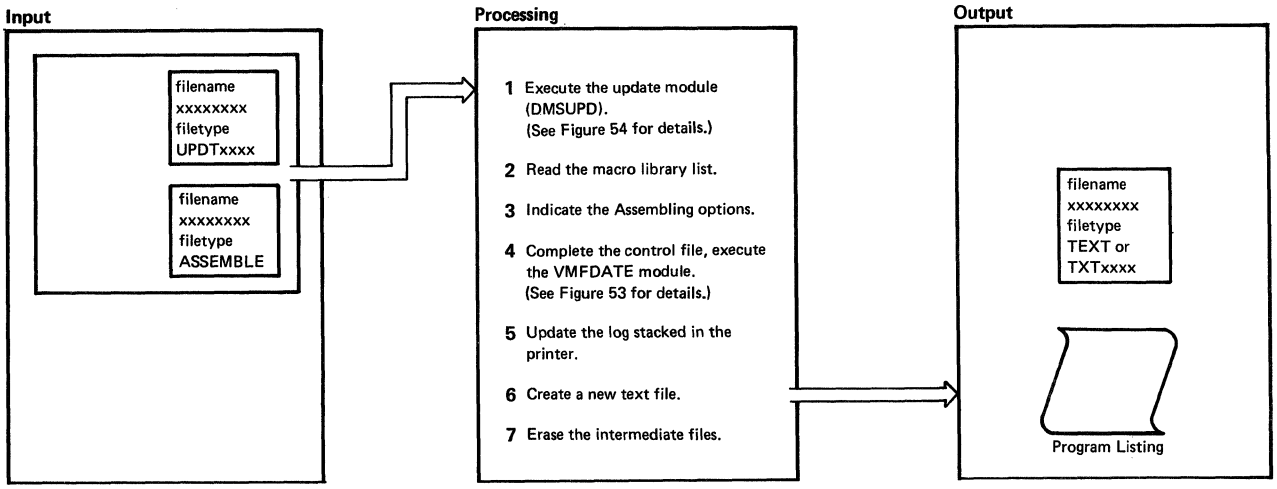


Figure 50. Overview of the Assembler Update Procedure



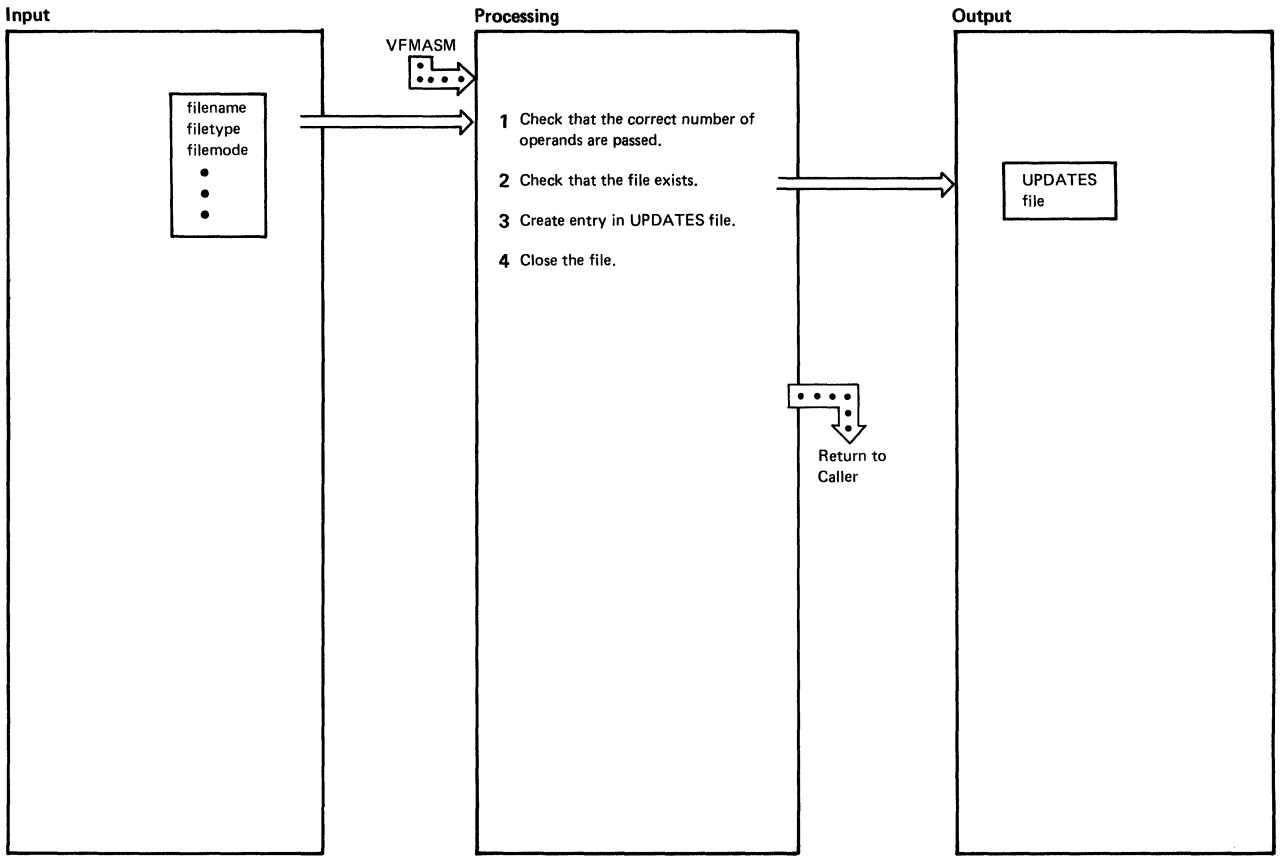
Notes		Module	Label	Ref
1	The CMS commands executed and the return codes that result will not be displayed on the virtual machine console.	VMFASM		
2	The default options are: PRINT, TERM, LIST, NODECK, NORENT, SYSPARM(SUP), XREF(SHORT), and NORLD. The options specified for the VMFASM EXEC are: DISK, NOTERM, NOLIST, DECK, RENT, EXP, XREF, and RLD.	VMFASM		
3	The CMS STATE command is executed. A nonzero return code indicates that the ASSEMBLE file was not found.	VMFASM	STSYS	
4	The CMS STATE command is executed. A nonzero return code indicates that the CNTRL file was not found.	VMFASM	STCTL	
5	The CP SPOOL command is executed.	VMFASM	FUPD	

Figure 51. Initialization of the VMFASM Procedure



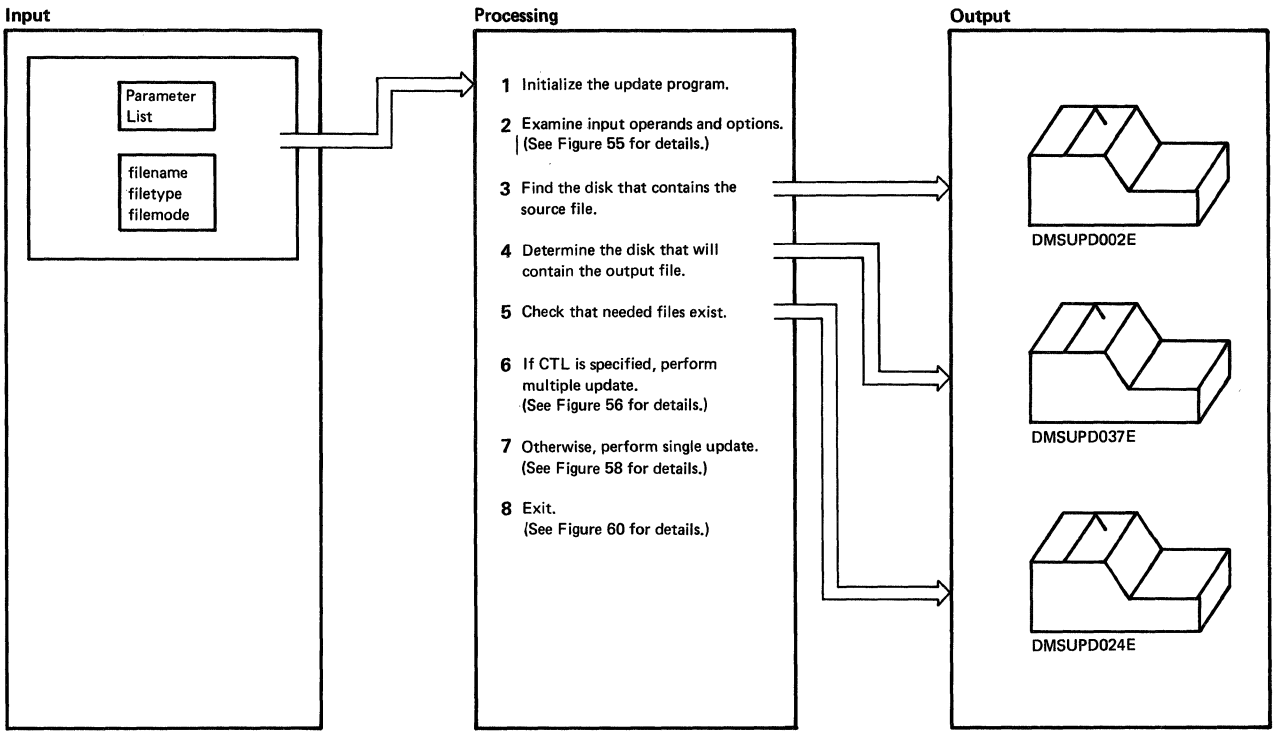
Notes		Module	Label	Ref
1	The DMSUPD module is executed. The name of the ASSEMBLE and CNTRL files and a filetype of ASSEMBLE are passed to the DMSUPD module. The DMSUPD module returns a level identifier and a MACLIB (macro library) list. A return code between 20 and 36 causes the VMFASM EXEC procedure to display the message ***ERROR UPDATING filename and return control to the CMS command environment. If the level identifier is TEXT, TEXT becomes the filetype of the completed text deck. If the level identifier (xxxxx) is not TEXT, the filetype becomes TXTxxxxx. If the return code is 40 (no updates), the filename is the same as the filename of the original ASSEMBLE file. Otherwise, the filename is set to the updated filename.	VMFASM	FUPD	
2	The MACLIB list is read. The VMFDATE module is executed once for each MACLIB. The CMS GLOBAL command is issued to identify the macro libraries that will be used during the assembly.	VMFASM		
3	If any options were specified on the VMFASM command, the message ASMBLING filename (options...) is displayed indicating the specified options	VMFASM		
	If no options were specified on the VMFASM command, the default options are assumed and the message ASMBLING filename is displayed.		ASMP	
4	The VMFDATE module is executed once more to complete the UPDATES file.	VMFASM	DTF	
5	The UPDATES file is printed on the virtual printer and then erased.	VMFASM	DTF	
6	The updated file is assembled. If ASSEMBLE returns a nonzero code, the message ***ERROR ASMBLING filename*** is displayed. The STATE command is issued to see if a text deck actually exists. If the text deck does not exist, the message ***NO TEXT FOR filename*** is displayed, the VMFASM EXEC procedure terminates, and control returns to the CMS command environment.	VMFASM	DTF	
7	The new text file, original ASSEMBLE file, and any UPDT xxxx files are saved. The message TEXT filename CREATED TXTxxxx is displayed. All intermediate files are erased. The printer is closed and control returns to the CMS command environment.	VMFASM	COMB EXIT	

Figure 52. Assembling Portion of the VMFASM Procedure



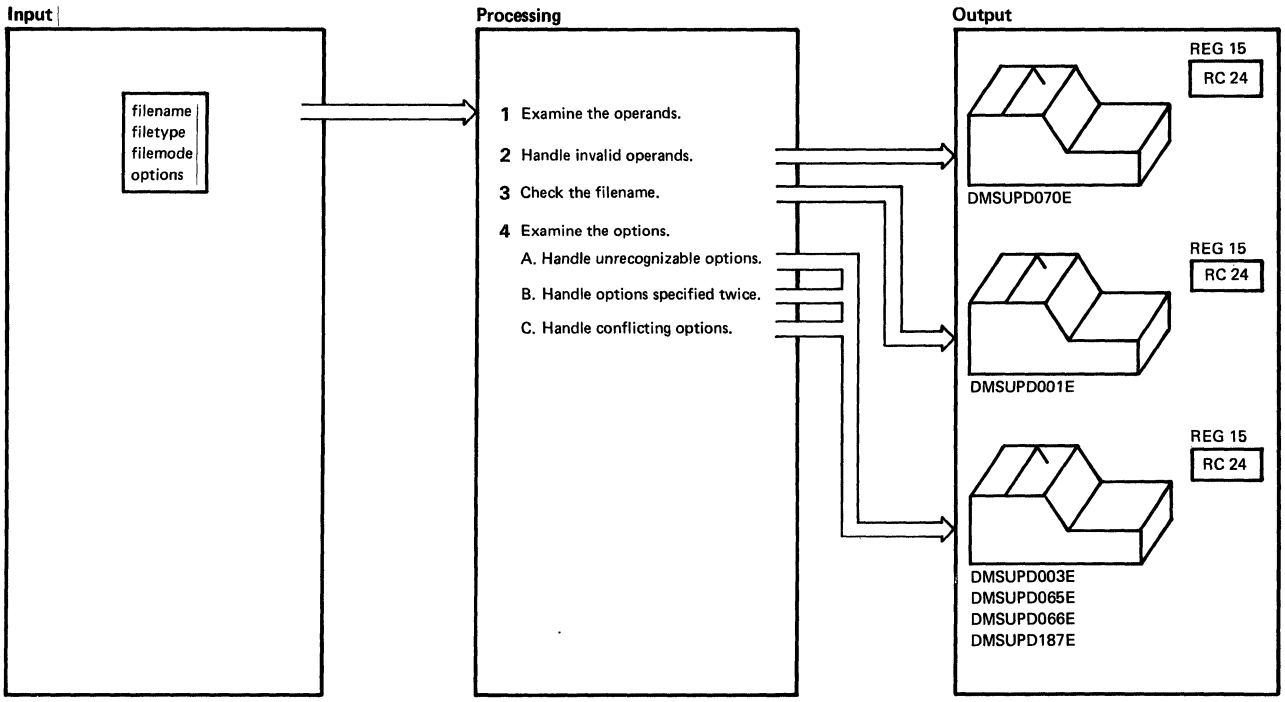
Notes		Module	Label	Ref
1	Six operands should be passed to the VMFDATE module. The first three operands are the filename, filetype, and filemode of the input file. The next three operands are the filename, filetype, and filemode of the output file.	VMFDATE	VMFDATE	
2	If the input file does not exist, control returns to the calling routine.	VMFDATE	TEST	
3	Each time the VMFDATE module is called, it creates an entry in the VMCNTRL file indicating that an update was applied. The format of each entry is: * filename filetype filemode volid date time The disk label is picked up from the ADT (Active Disk Table).	VMFDATE		
4	The UPDATES file is closed and control returns to the calling routine.	VMFDATE		

Figure 53. VMFDATE Program



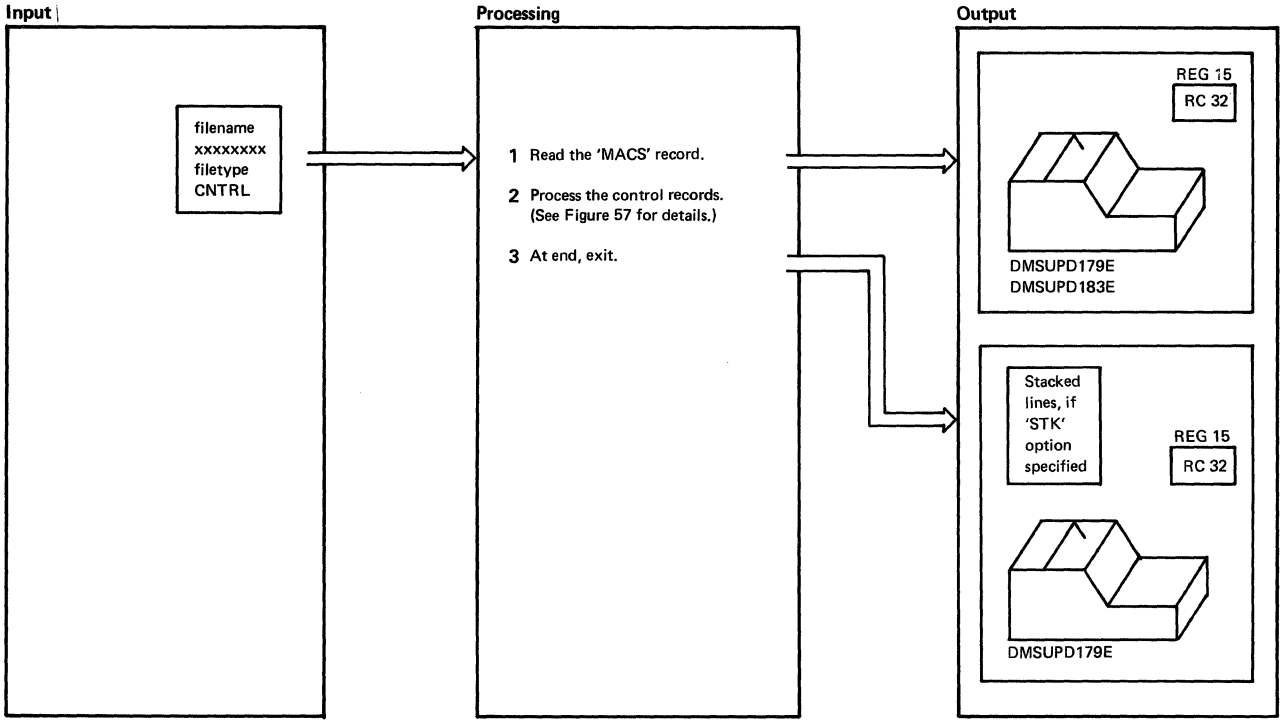
Notes		Module	Label	Ref
1	Registers 12, 11, and 9 are set up as base registers. All indicators are set off.	DMSUPD	DMSUPD	
2	The filename operand is required.	DMSUPD	DMSUPD	
3	DMSUPD checks that the source input file exists. If not, the message DMSUPD002E File [*fn [ft [fm]]*] not found is displayed and control returns to the CMS command environment with a return code of 28 in register 15.	DMSUPD	PROCESS NOFILE	
4	The DMSUPD module searches for a suitable disk to hold the output files. First, an attempt is made to place the files on the same disk that contains the original input. If the input disk is read-only, but is an extension of a read/write disk, an attempt is made to place the files on that disk. Lastly, an attempt is made to place the files on the A-disk. If all these attempts fail, the message DMSUPD037E Disk *mode*[(*vdev*)] is accessed as read/only is displayed and control returns to the CMS command environment with a return code of 36 in register 15.	DMSUPD	PROCESS	
5	DMSUPD issues the STATE command to see if the UPDATE CMSUT1 file already exists; it should not exist. If the CMSUT1 file exists, the message DMSUPD024E File *fn ft fm* already exists is displayed and control returns to the CMS command environment with a return code of 24 in register 15.	DMSUPD	PROCESS	
	If the DISK option was specified, an old copy of 'filename UPDLOG' is erased (if one exists).			
	If the control file option (CTL) is specified, DMSUPD checks that the control file exists and continues processing at the CTLMUTL (multiple update) routine.		NOERASE	
	If the control file option is not specified, DMSUPD checks that the single update file exists and continues processing at the single update (SINGUPD) routine.		LOCTUPD	
6	See Figure 56.	DMSUPD	CTLMULT	
7	See Figure 59.	DMSUPD	SINGUPD	
8	See Figure 60.	DMSUPD	RETR001	

Figure 54. Overview of the Update (DMSUPD) Program)



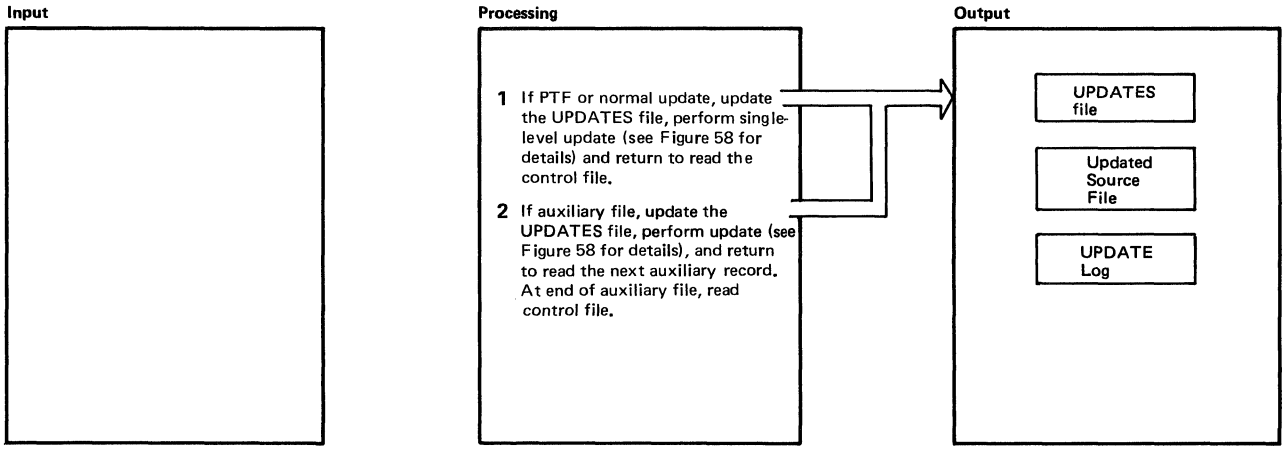
Notes	Module	Label	Ref
1 DMSUPD uses the filename operand to set up the disk parameter lists for input, update log, and auxiliary files. All the operands (except the required filename) and all the options are read by branching and linking to the OPTSCAN routine. The first three operands are the filename, filetype, and filemode of the file to be updated. The next three operands are the filename, filetype, and filemode that describe the update or control file to be applied.	DMSUPD	DMSUPD	
2 If more than six operands are specified before the left parenthesis, the message DMSUPD070E Invalid parameter *parameter* is displayed and control returns to the CMS command environment with a return code of 24 in register 15.		EXCESIV	
3 Only the first operand must be specified. If no operands are found, the message DMSUPD001E No filename specified is displayed and control returns to the CMS command environment with a return code of 24 in register 15.	DMSUPD	NOFNAME	
4 The options assumed, if not otherwise specified are: SEZ8, NOINC, NOREP, NOCTL, NOSTK, TERM, and DISK. When the last option is processed, control returns to the PROCESS routine.	DMSUPD		
A. If an unrecognizable option is specified, the message DMSUPD003E Invalid option: *option* is displayed and control returns to the CMS command environment with a return code of 24 in register 15.		INVOPTN	
B. If an option is specified twice, the message DMSUPD065E *option* option specified twice is displayed and control returns to the CMS command environment with a return code of 24 in register 15.		OPTDUP	
C. If two conflicting options are specified, the message DMSUPD066E *option1* and *option2* are conflicting options is displayed and control returns to the CMS command environment with a return code of 24 in register 15. The conflicting pairs of options are: SEQ8, and NOSEQ8, INC and NOINC, REP and NOREP, STK and NOSTK, TERM and NOTERM, CTL and NOCTL, INC and NOINC, and DISK and PRINT.		OPTCONF	
If the STK option is specified without the CTL option, the message DMSUPD187E Option STK invalid without CTL is displayed, and control returns to the CMS command environment with a return code of 24 in register 15.		ERSC	

Figure 55. Operand and Option Checking



Notes		Module	Label	Ref
1	The macro library (MACS) record is read from the beginning of the control file and saved. If the MACS card is not found, or is not the first noncomment card in the control file, the message DMSUPD179E Missing or duplicate MACS card in control file *fn ft fm* is displayed and control returns to the CMS command environment with a return code of 32 in register 15.	DMSUPD	CTLMULT ERMACS	
	If the MACS control card is invalid, the message DMSUPD183E Invalid {CONTROL|AUX} file control card is displayed and control returns to the CMS command environment with a return code of 32 in register 15.		BADCTLC	
2	See Figure 57.	DMSUPD	CTLREST	
3	If a 'MACS' record is read, the file is completely processed. The control file is closed.	DMSUPD	CTLDONE	
	If this MACS card does not have an item number identical to that of the MACS control card originally read, the control file contains duplicate MACS control cards. The message DMSUPD179E Missing or duplicate MACS card in control file *fn ft fm* is displayed and control returns to the CMS command environment with a return code of 32 in register 15.		ERMACS	
	If STK is specified, the updated level ID is stacked in the terminal input stack.			

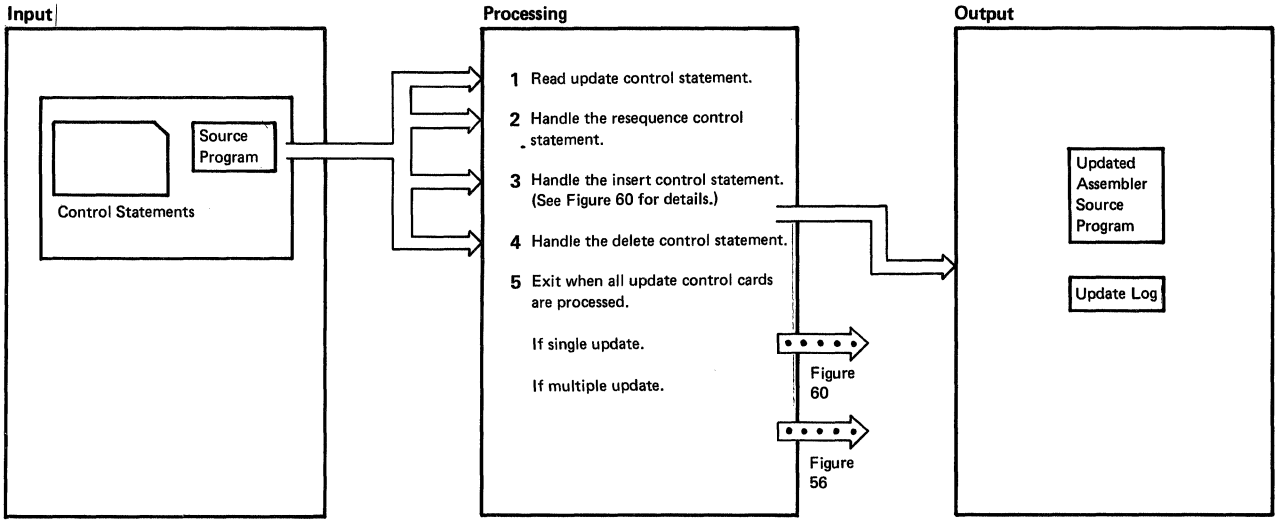
Figure 56. Multiple Update Procedure



Notes		Module	Label	Ref
1	The control file is read from the bottom up. If the control record is valid, the message DMSUPD183E Invalid {CONTROL|AUX} file control card is displayed and control returns to the CMS command environment with a return code of 32 in register 15.	DMSUPD	CTLREST CTLREAD BADCTLC	
	If the PTF or update file is not found, control returns to the read routine (CTLREAD). If the file is found and the update is not being performed in storage, the message DMSUPD178I Updating *fn ft fm* is displayed and an entry is made in the UPDATES file. If the update is being performed in storage, free storage is acquired to contain the input file. The message DMSUPD300E Insufficient storage to begin update is displayed if the input file is too large for the acquired storage. If the STOR option was not specified explicitly, the message DMSUPD304I Update processing will be done using disk is also displayed. If the STOR option was specified, control returns to CMS with a return code of 40 in register 15. If processing continues, the input file is read into the acquired storage, the message DMSUPD178I Updating *fn ft fm* is displayed, and an entry is made in the UPDATES file. Then a branch to the SIGNUPD routine transfers control to the single update routine. After the update is performed, control returns to CTLCONT.		CTLIPTF CTLOCUP CTLUMSG CTLUMSS SMALLCOR IMPLICIT CTLUMSS	
2	DMSUPD checks that the auxiliary file exists. If not, control returns to the read routine (CTLREAD). If the auxiliary file is found, it is read from the bottom up. If the PTF file within the auxiliary file is not found, the message DMSUPD180W Missing PTF file *fn ft fm* is issued to the console and written to the 'fn' UPDLOG. The RETCODE value is set to 12 if it has not been set higher previously. Processing continues with the next record from the auxiliary file (AUXREAD). When a valid record is read from the auxiliary file, the message DMSUPD178I Updating *fn ft fm* is displayed and an entry is made in the UPDATES file. Then the SINGUPD routine applies the update. After the update is performed, control returns to CTLCONT which returns control to AUXREAD. This loop continues until the entire auxiliary file is processed. At the end of the auxiliary file, the file is closed and control returns to the control file read routine (CTLREAD). If an invalid card is found in the auxiliary file, the message DMSUPD183E Invalid {CONTROL|AUX} file control card is displayed and control returns to the CMS command environment with a return code of 32 in register 15.	DMSUPD	AUXFIND NOFILEW CTLUMSG CTLUMSS AUXREAD AUXFINT BADAUXC	

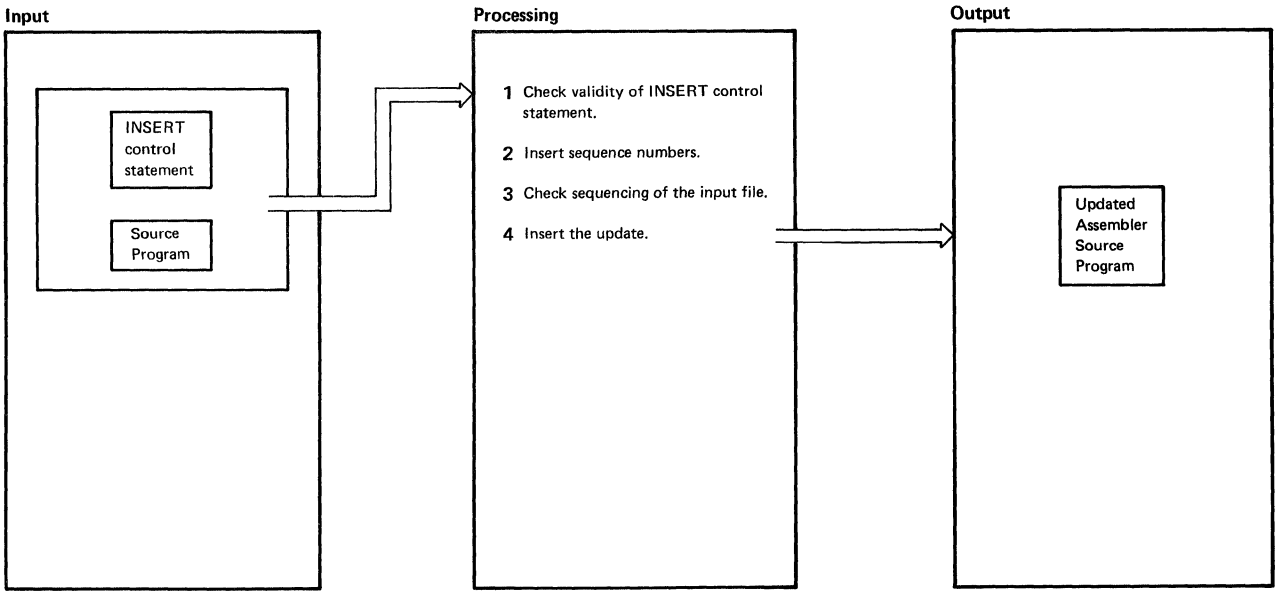
Figure 57. Control Record Processing

Generating and Updating VM/SP



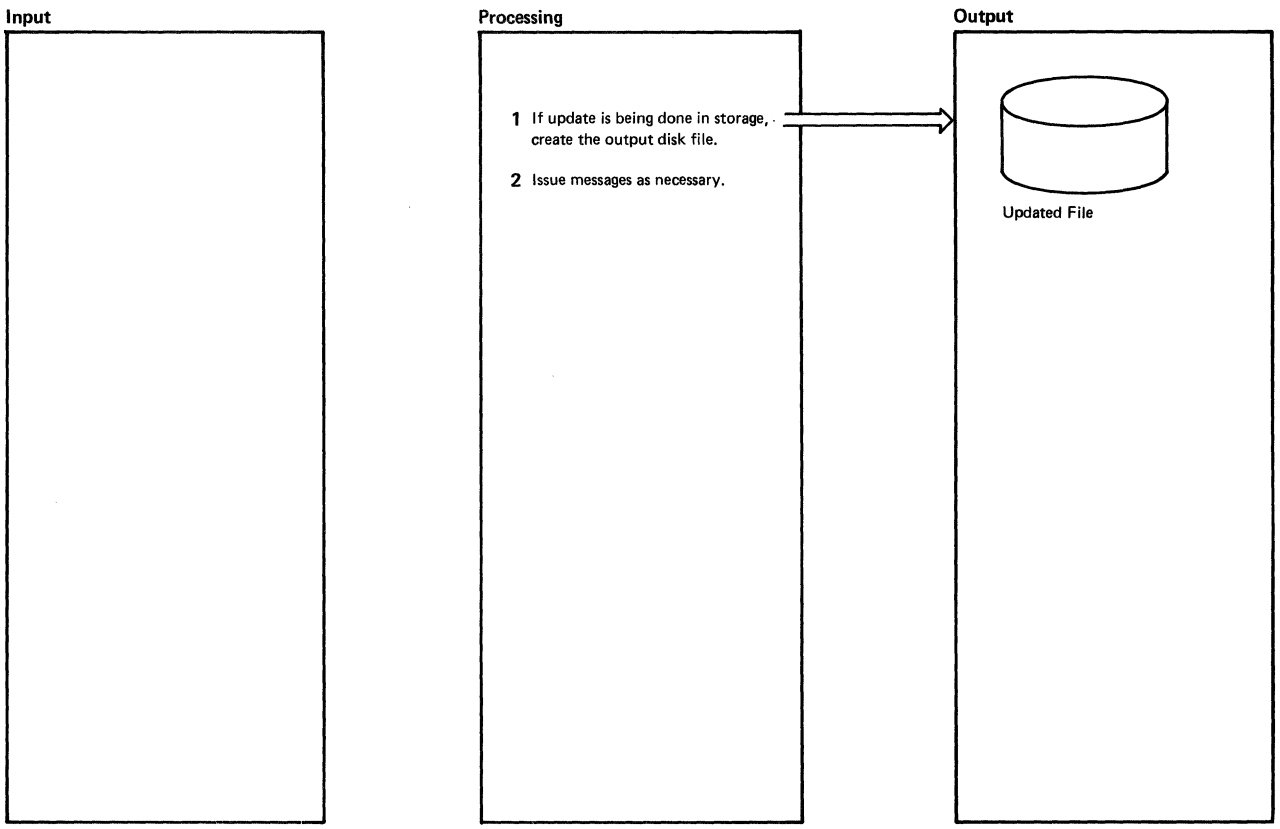
Notes		Module	Label	Ref
1	An update card is read and checked, if an invalid control card is read, the message DMSUPD207W Invalid update file control card is issued. The value of RETCODE is set to 12, if it was not previously set higher. Processing continues ignoring the invalid card.	DMSUPD	SINGUPD	
2	DMSUPD checks the resequence card. If the resequence card is not the first card in the update file, the message DMSUPD184W ./S not first card in update file--ignored is issued. The value in RETCODE is set to 12 if it has not been set higher previously. The './S' card is ignored and processing continues.	DMSUPD	FCTRSEQ RSEQERR	
	If an invalid character is specified in one of the sequence fields, the message DMSUPD185W Invalid character in sequence field *seqno* is issued. The value of RETCODE is set to 12 if it was not set higher previously. The './S' card is ignored and processing continues.		INVCHAR	
	If the specified sequence increment is zero, the message DMSUPD182W Sequence increment is zero is issued. The value of RETCODE is set to 8 if it has not been set higher previously. Processing continues and the file is resequenced with a sequence increment of zero.		ZERSEQ	
	If no errors are found, the sequencing is set to 5 or 8 characters depending on the options specified (SEQ8 or NOSEQ8). The UPDFLAG is set for resequencing and the next update control card is read (UPREAD).		RETR001 RSEQFIN	
3	See Figure 60.	DMSUPD	RCTINST	
4	The update control card is checked. The indicated cards are removed. The control statement and the message DELETING... are sent to the UPDLOG file. If the delete is being performed in storage, the records in storage are reclaimed, eliminating the deleted records.	DMSUPD	FCTDELT DELTINE XDELE	
5	When all the update control cards are processed, the UPDREAD (read) routine takes its error exit (UPDFERR). The UPDFERR routine branches to the INPUTRD routine on and end-of-file condition to flush (write out) the rest of the input source file is the update was not performed in storage. If the update was performed in storage, and resequencing is requested, a logical replace is done on each line in the file.	DMSUPD	UPDREAD XDELE	
	The error exit (INPFERR) is taken from the INPUTRD routine. The INPFERR routine closes the updated file and the input file. If processing a control file (multiple update), control returns to CTLCONT. Otherwise, the single-level update is complete and control is returned to CMS (RRETURN routine).			

Figure 58. Single Update Procedure



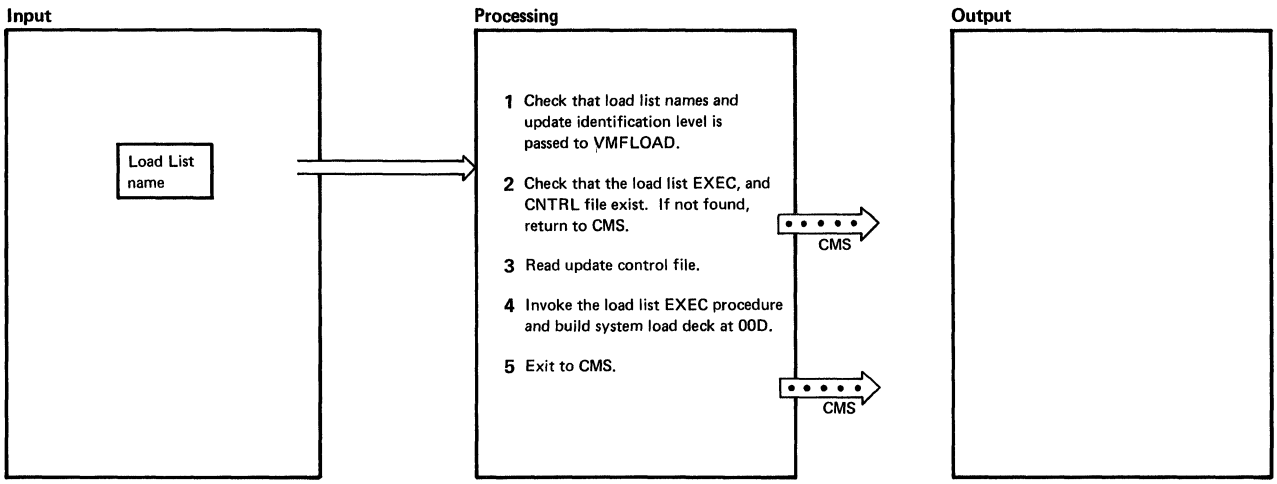
Notes		Module	Label	Ref
1	The INSERT card is checked. If invalid, the message DMSUPD207W Invalid update file control card is issued. The value of RETCODE is set to 12 it if was not set higher previously. The invalid card is ignored and processing continues.	DMSUPD	FCTINST INVUPCD	
2	If requested, the sequence numbers are put in the inserts. Otherwise, the sequence number field contains ******** If a specified sequence number is not found, the message DMSUPD186W Sequence number *seqno* not found is issued. The value of RETCODE is set to 12 if it has not been set higher previously. The invalid card is ignored and processing continues.	DMSUPD	FCTREPL UPDSERR	
3	If the input file sequence numbers are out of order, the message DMSUPD210W Input file sequence error: *seqno1* to *seqno2* is issued. The value of RETCODE is set to 4 if it was not set higher previously. Processing continues.	DMSUPD	INSEQW	
4	DMSUPD inserts the cards. The control statement and the INSERTING... message are sent to the 'UPDLOG' file. If the sequence errors are introduced in the coutput file, the message DMSUPD174W Sequence error introduced in output file: *seqno1* to *seqno2* is issued. The value of RETCODE is set to 8 if it was not set higher previously. Processing continues. If sequence overflow occurs while cards are being inserted, the message DMSUPD176W Sequencing overflow following sequence number *seqno* is issued. The value of RETCODE is set to 8 if it was not previously set higher. Processing continues. When the appropriate cards are successfully inserted in the file, control returns to the read routine to read the next control card.	DMSUPD	INSLOOP WOVF	

Figure 59. Inserting Updates



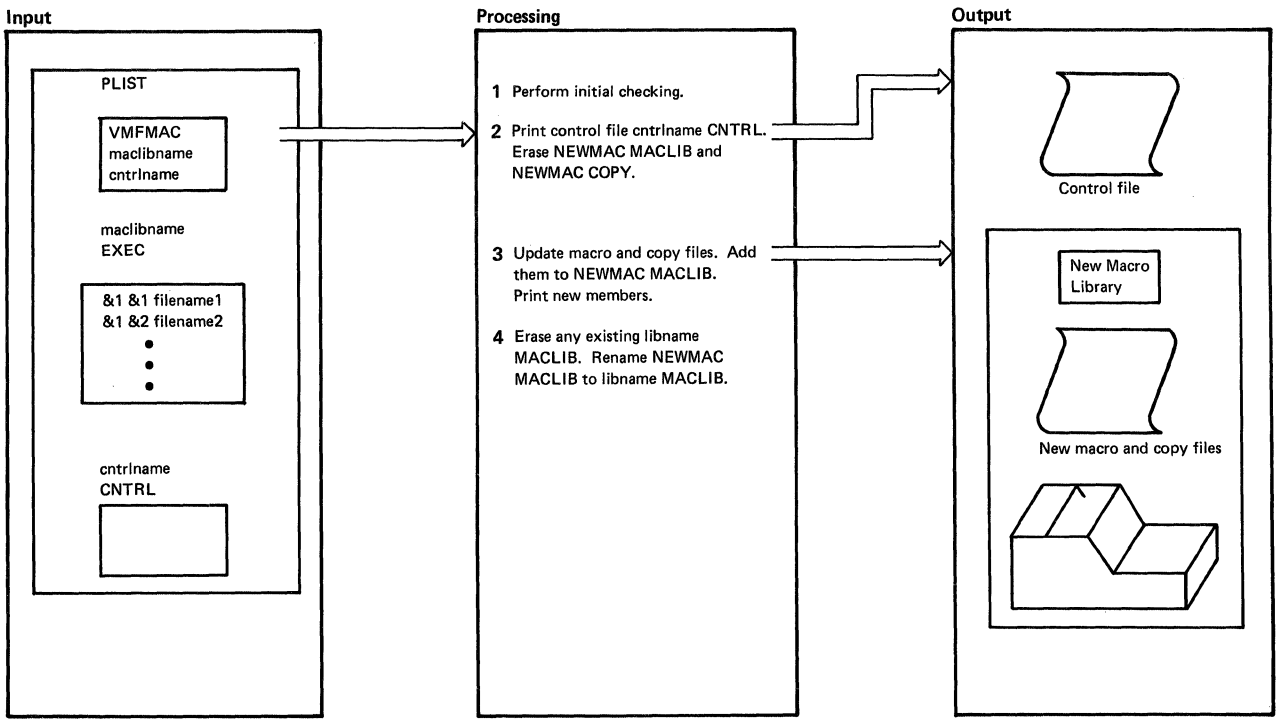
Notes		Module	Label	Ref
1	If the update is being performed in storage, the updated file in storage is read line by line and a disk file is created with the filename and filetype UPDATE CMSUT1. The filemode specifies the disk where the final output file resides. The disk file is then closed. The UPDATE CMSUT1 file is then renamed $fname after the old $fname is erased.	DMSUPD	RETR001 RETRD	
2	If RETCODE is not equal to zero, warning messages were issued during the update.	DMSUPD		
	If warning messages are issued and the NOTERM option is specified, while the REP option is not, the message DMSUPD177I Warning messages issued (severity = nn)&sbrk.;REP option ignored] is displayed (nn is the value in RETCODE).		WRETURN	
	If warning messages are issued and the REP option is specified, the message DMSUPD177I Warning messages issued (severity = nn)&sbrk.;REP option ignored] is displayed (nn is the value of RETCODE). In either case, control returns to the CMS command environment with the value of RETCODE in register 15.			
	If no warning messages are issued and the REP option is specified, the '$fname' file is renamed to 'fname', after the old file is erased.			
	If the CTL option is specified and no update files are found, the message DMSUPD181E No update files were found is displayed and control returns to the CMS command environment with a return code of 40 in register 15.		NOUPDATS	
	If no warning messages are issued, and no errors detected, control returns to the CMS command environment with a return code of 0 in register 15.			

Figure 60. Exit Processing



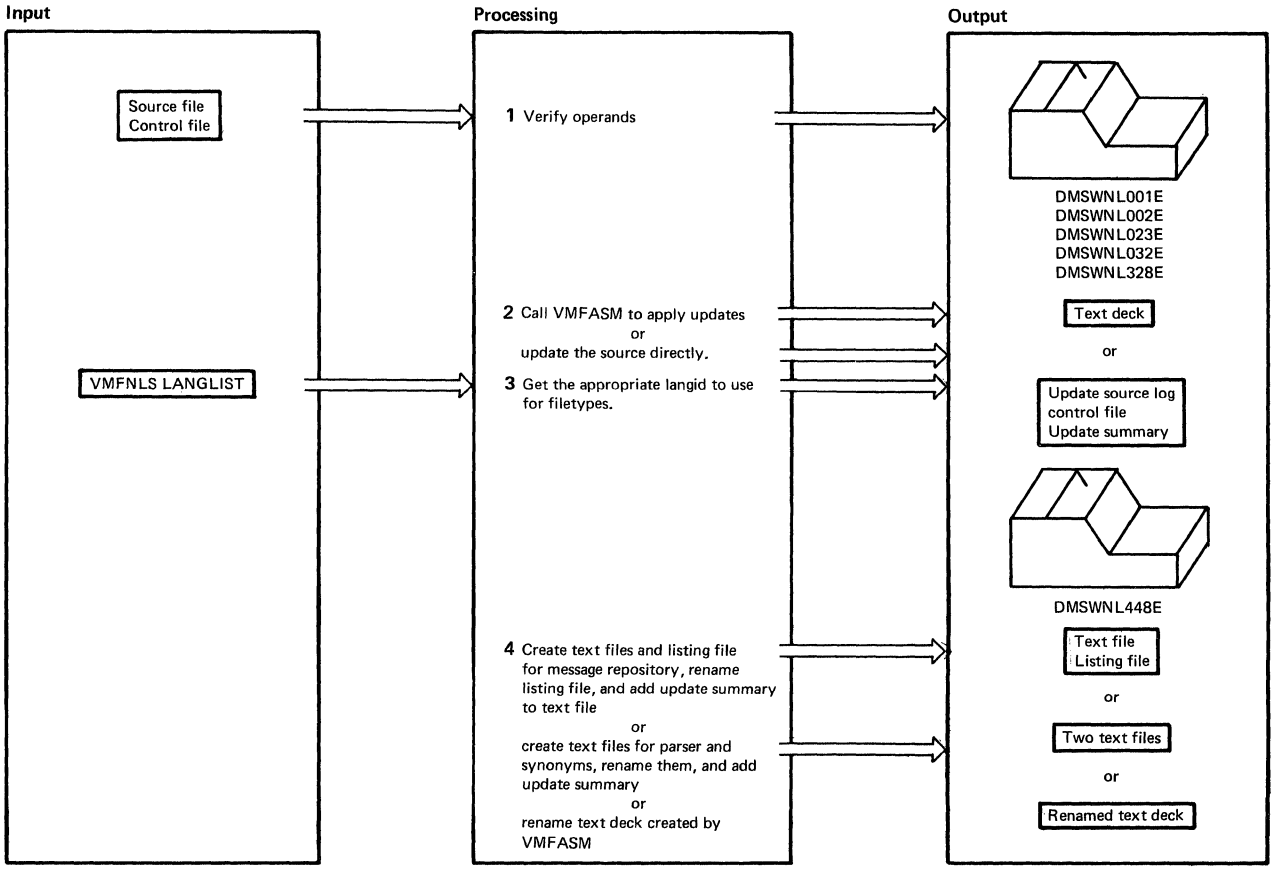
Notes		Module	Label	Ref
1	The load list name is moved into the filename portion of a STATE command line for an EXEC file and a CNTRL file.	VMFLOAD	VMFLOAD	
	The update identification level is moved into the filename portion of a RDBUF command line for an EXEC and CNTRL file.			
	The langid is retrieved from the command line if entered.	VMFLOAD	NOLDL	
2	Issue the STATE command via an SVC 202 to make sure that the load list EXEC and CNTRL files exist. If the load list EXEC file is not found, the message NO LOAD LIST is displayed and control returns to the CMS command environment with a return code of 4 in register 15.		NOCTR	
	If the load list CNTRL file is not found, the message NO CONTROL FILE is displayed and control returns to the CMS command environment with a return code of 2 in register 15.	VMFLOAD	DINITA	
3	The first record of the control file is read and the class on the macro library record is saved.		RDCTR	
	The rest of the control file is read. The control records are chained together in the proper hierarchy.		BDCTR	
	If an error occurs while reading the control file, the message ERROR IN CONTROL FILE is displayed and control returns to the CMS command environment with a return code of 3 in register 15.	VMFLOAD	DINITB	
4	The punch is set to spool continuously. The load list EXEC procedure is invoked by an SVC 202. The text files are punched in the order specified in the load list.		NOFILE	
	The resident nucleus modules are loaded first and the pageable modules follow. The DMKLD00E (nucleus loader) resident nucleus module must be loaded first and followed by DMKPSA. The DMKCPE module must be the last resident nucleus module loaded. The pageable nucleus modules are ordered so that they efficiently utilize page frames. The DMKSAV module must be loaded last. When the filename and filetype are both specified, that specific file is searched for and punched, if found. When (LANG is specified, the file filename TXTlangid is searched for and punched. If the langid is not specified, then LANG is ignored. If the file is not found, it is skipped, the message filename filetype NOT FOUND is displayed, and processing continues with the next item in the load list.		FNDM DINITD SRTXT NOFILE	
	When only the filename is specified, the specified control file is used to search for the highest level text file available. The first text file located is punched. If the search ends before a text file is found, the "filename TEXT" file is punched if it exists. If the file is not found, it is skipped, the message filename filetype NOT FOUND is displayed, and processing continues with the next item in the load list.			
	This process continues until every item in the load list is processed.	VMFLOAD	ENDL	
5	At this point, the text decks are loaded in the proper sequence in the specified reader. All files not found were identified by messages to the terminal. The message SYSTEM LOAD DECK COMPLETE is displayed. The punch is set to stop spooling and is then closed.		RETERR	
	Control returns to the CMS command environment.			

Figure 61. VMFLOAD Program



Notes		Module	Label	Ref
1	If a list of the members to be put in the macro library (maclibname EXEC) is not found, the message maclibname EXEC NOT FOUND is displayed and control returns to CMS with a return code of 101.	VMFMAC	ASSGN	
	If the file containing the updates is not found, the message cntrlname CNTRL NOT FOUND is displayed and control returns to CMS with a return code of 102.		STCTL	
2	The control file cntrlname CNTRL is printed. The files NEWMAC MACLIB and NEWMAC COPY are erased.	VMFMAC	STKL	
3	If a macro or copy file is not found, the message ***filename COPY OR MACRO NOT FOUND*** is displayed. The final return code is set to 104 and processing continues with the next member.	VMFMAC	AREAD	
	The UPDATE command is issued for each macro or copy file. If an error occurs, the message ***ERRORS UPDATING membername membertype*** membername membertype NOT INCLUDED IN MACLIB is displayed on the terminal, the files membername UPDATES and membername UPDATES and membername membertype are printed. The final return code is set to 105 and processing continues with the next member.		MACUP UPDERR	
	If the update procedure is successful, VMFDATE is executed to data stamp the file, and the member is added to the NEWMAC MACLIB. The new member is printed. To maintain a history of the updates that were applied, a line is added to NEWMAC COPY, a dummy copy file.		MACUP	
4	After all macro and copy files have been processed, the NEWMAC COPY file is renamed to libname COPY and added to NEWMAC MACLIB. Any existing libname MACLIB file is erased and the NEWMAC MACLIB is renamed to libname MACLIB.		RENEWCO	
	If the update procedure is unsuccessful, the message DUE TO PREVIOUS ERRORS, THE RESULT OF THE MACLIB BUILD IS CALLED 'NEWMAC MACLIB' libname MACLIB HAS NOT BEEN REPLACED is displayed at the terminal and a return is made to CMS with the final return code as previously described.		ERR2	

Figure 62. VMFMAC--The Macro Library Creation Procedure

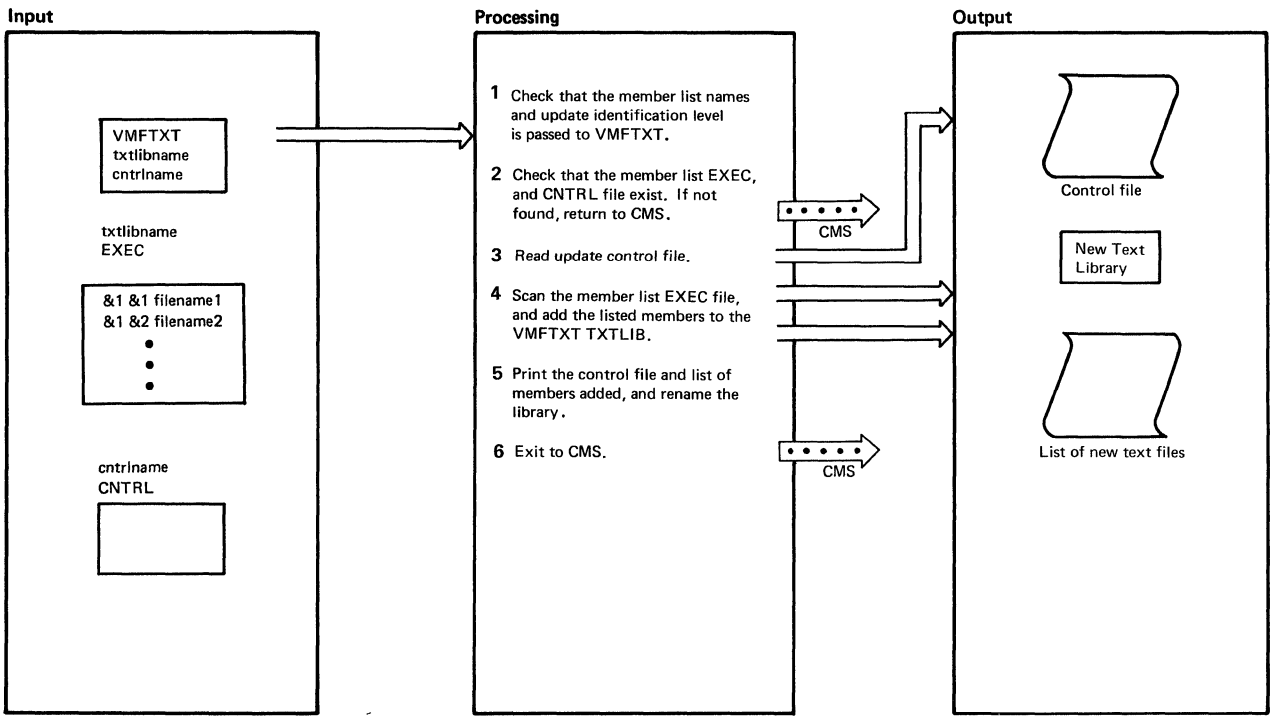


Notes		Module	Label	Ref
1	Checks that: • A source filename was specified; if it was not, issue the message DMSWNL001E FILENAME NOT SPECIFIED • A source filetype was specified; if it was not, issue the message DMSWNL023E FILETYPE NOT SPECIFIED • A control file was specified; if it was not, issue the message DMSWNL328E CONTROL FILE NOT SPECIFIED • The source filetype is either REPOS, DLCS, or ASSEMBLE; if it was not, issue the message DMSWNL032E INVALID FILETYPE *ft*	VMFNLS		
2	The source file, control file, and VMFNLS LANGLIST file all exist; if one (or more) of them don't, issue the message DMSWNL002E FILE *fn ft* NOT FOUND			
2	If the source filetype is ASSEMBLE, call VMFASM to apply the updates. If the source file is DLCS or REPOS, do the following: • Update the input source file and print update source log If the update fails, quit the program. • Print control file and update summary file.			
3	If the source file contains a country code in the 7th and 8th characters of its filename, use this code to search a file called VMFNLS LANGLIST for the appropriate langid; If the country code is not found in VMFNLS LANGLIST, issue the message DMSWNL448E COUNTRY CODE *code* NOT IN VMFNLS LANGLIST If the source filename only has 6 characters, the langid defaults to AMENG.			

Figure 63 (Part 1 of 2). VMFNLS -- Updating National Language Files

Notes	Module	Label	Ref
4 If the source filetype is REPOS, do the following: ● Create the message repository text file by issuing GENMSG. (Quit the program if GENMSG fails.) ● Rename the GENMSG listing file so it matches the filename of the input source file, and print the listing. ● Append the summary of updates to the front of the text file, then erase the summary of updates.			
5 If the source filetype is DLCS, do the following: ● Create the parser and synonym text files by issuing CONVERT COMMANDS. (Quit the program if CONVERT COMMANDS fails.) ● Rename the text files so they match the filename of the input source file, and print them. ● Append the summary of updates to the front of the text file, then erase the summary of updates.			
6 If the source filetype is ASSEMBLE, rename the text deck created by VMFASM.			

Figure 63 (Part 2 of 2). VMFNLS -- Updating National Language Files



Notes	Module	Label	Ref
1 Checks that: • The message issuing routine exists. If the file cannot be found, issue the message: DMSWTX002E File *fn ft fm* not found • There are no leftover work files. If the file already exists, issue the message: DMSWTX024E File *fn ft fm* already exists • All required system files exist. If the file(s) cannot be found, issue the message: DMSWTX002E File *fn ft fm* not found • The invocation is correct. If no filename is specified, issue the message: DMSWTX001E No filename specified If the format is incorrect, issue the message: DMSWTX026E Invalid parameter *parameter* for *function* function Verifies that: • The given file names are valid. If they are not, issue the message: DMSWTX062E Invalid character *char* in fileid *fn ft* • The specified files exist. If the file cannot be found, issue the message: DMSWTX002E File *fn ft fm* not found • the A-disk is R/W. If the A-disk is not R/W, issue the message: DMSWTX006E No read/write A disk accessed	VMFTXT	MAIN	

Figure 64 (Part 1 of 2). VMFTXT -- The Text Library Creation Procedure

Notes		Module	Label	Ref
	Builds the list of filetypes to be searched. Determines the number of records in the member list. Loops through the entries in the member list. Blank lines, lines beginning with an *, &CONTROL or &TRACE, and words beginning with an & are ignored. The remaining entries are checked for valid format and content. If the entry is not valid or a minor error occurs while trying to add any member, issue the message: DMSWTX056E File *fn ft [fm]* contains invalid record formats and continue processing the remaining members. If a major error has occurred, quit. Print the CNTL file (if used), and the library contents file (VMFTXT TEXT). Issue the message: DMSWTX895I Member *fn ft* added Add the library contents member to the library. If there have been no errors, erase any existing A-disk copy of the txtlib and rename VMFTXT TXTLIB to the requested name. If there were any errors, issue the message: DMSWTX897E Due to previous errors, the result of this TXTLIB build is called VMFTXT TXTLIB; your *fn* TXTLIB has not been replaced.			
2	Verify that the specified filename/filetype does in fact exist. If found, call ADD_TXT. If not found, issue the message: DMSWTX896E File *fn ft fm* not found.	VMFTXT	WITH_TYPE	
3	Search through the list of filetypes defined in the user specified CNTRL file. If found, call ADD_TXT. If not found, issue the message: DMSWTX896E File *fn* TEXT or *fn* TXT* not found	VMFTXT	FIND_TYPE	
4	If the given filetype is not TEXT and there is a copy of the given file on the A-disk with a filetype of TEXT, then rename the existing TEXT file to VMFTXT CMSUT1 A. Record the original name in VMFTXT CMSUT2 A. If the given filetype is not TEXT, make a copy of the given file on the A-disk with the given filename and a filetype of TEXT. Add the specified file to the VMFTXT TXTLIB. Also, add the date/time stamp information on the new member to the VMFTXT TEXT file. Issue the message: DMSWTX895I Member *fn ft* added If the given filetype is not TEXT, erase the temporary TEXT file from the A-disk, and restore any previous copy the user may have.	VMFTXT	ADD_TXT	
5	If a CTL filename was NOT given on the VMFTXT command, the search list consists of 1 filetype, which is TEXT. Otherwise ... Stack the entire CNTRL file for processing. Determine how many entries are on the stack. Read in the stacked lines, ignoring blank lines or lines beginning with *. If the first token on the line is not TEXT, add a prefix of TXT. Verify that the first record is a MACS record. Check for any extra MACS records. If an error, issue the message: DMSWTX179E Missing or duplicate MACS card in control file *fn ft fm* Verify that all entries are valid filetypes. If there's an error, issue the message: DMSWTX183E Invalid control file control card	VMFTXT	CNTRL	
6	Check whether the filename/filetype parameter contains characters other than those that are valid for a CMS file identifier. Check whether the filename/filetype parameter is longer than 8 characters.	VMFTXT	VERFN	

Figure 64 (Part 2 of 2). VMFTXT -- The Text Library Creation Procedure

Program Organization

The VM/SP procedures for generating and updating VM/SP consist of VMFASM, VMFNLS, VMFMAC, VMFTXT, VMFDATE, DMSUPD, and VMFLOAD.

The Assembler language update procedure consists of the VMFASM EXEC procedure and two modules (VMFDATE and DMSUPD). The VMFASM EXEC procedure sets up for the assembly by calling DMSUPD to create the update control file. There is an entry in the VM/SP Control file for each update control and auxiliary update file. The VM/SP Control identifies the updates applied to the original assembler program and the date and time they were applied.

The Assembler language update procedure calls the VMFDATE program. The MACLIBs needed are then included in the VM/SP Control file.

The nucleus loader procedure consists of a program (VMFLOAD) and an EXEC procedure. Although the DMSUPD update program is not used, the control file that it creates may be used. The VM/SP Load List procedure shows the nucleus modules in the order they are to be loaded. The list includes the filename of each module and may optionally include the update level. If the update level is not specified, the control file created by DMSUPD is used to locate the highest level update available, and that level of the module is loaded.

When nucleus modules are updated and loaded, it is often necessary to create a new macro library. The level of macro library needed for each updated module is recorded in the VM/SP Control file created by the VMFDATE module. The VMFMAC EXEC procedure creates a new macro library. The VMFTXT EXEC procedure rebuilds a TXTLIB file. A member list EXEC file contains the filenames and optional filetypes of the members to be included. For those members that do not specify a filetype, a list of filetypes (provided in the CNTRL file) is searched. This search processing is consistent with the output file created by the VMFASM EXEC procedure using the same CNTRL file.

Directory

Four label directories are provided:

1. The label directory for the Assembler update function, including labels from:
   - The VMFASM EXEC procedure
   - The DMSUPD update program
   - The VMFDATE control file program.
2. The label directory for the nucleus load program, VMFLOAD
3. The label directory for the VMFMAC EXEC. procedure, which creates and updates the macro library.
4. The label directory for the VMFTXT procedure which builds the text library.

Assemble Update Procedure Label Directory

Label	Module or Procedure	Figure	Description
-ASMP	ZVMFASM	52	Assumes default options for Assembler
AUXFINT	DMSUPD	56	Closes the auxiliary file when it is completely processed
AUXREAD	DMSUPD	56	Reads auxiliary file from the bottom up
BADAUXC	DMSUPD	56	Processing when invalid card found in auxiliary file
BADCTLC	DMSUPD	56	Abnormally terminates when an invalid control card is encountered
-COMB	VMFASM	52	Saves the new text file, original ASSEMBLE file, and UPDTxxxx files.
CORBUST	DMSUPD	59	Insufficient storage to complete update
CTLDONE	DMSUPD	56	Closes the control file once it is processed
CTLGETM	DMSUPD	56	Searches for first control card
CTLGOT1	DMSUPD	56	Checks that auxiliary file exists
CTLIPTF	DMSUPD	56	Checks that PTF file exists
CTLMULT	DMSUPD	54, 56	Multiple update processing
CTLOCUP	DMSUPD	56	Checks that update file exists
CTLREAD	DMSUPD	56	Reads the control file from the bottom up
CTLUMSG	DMSUPD	56	Updates the UPDATES file
CTLUMSS	DMSUPD	56	Issues the short update message
DELTINE	DMSUPD	58	Deletes cards from the source file
DMSUPD	DMSUPD	55	Entry to update program

Label	Module or Procedure	Figure	Description
-DTF	VMFASM	52	Stacks control file in printer
ERMACS	DMSUPD	56	Processing when MACS card invalid or missing
ERSC	DMSUPD	55	Processing when STK option specified without CTL option
EXCESIV	DMSUPD	55	Error exit when too many parameters are specified
-EXIT	VMFASM	52	Erases intermediate files and returns to CMS
FCTDELT	DMSUPD	58	Checks the delete control card for validity
FCTINST	DMSUPD	58, 59	Checks the validity of the insert control card
FCTREPL	DMSUPD	59	Checks the validity of the replace control card
FCTRSEQ	DMSUPD	58	Checks the resequence control card
-FUPD	VMFASM	51, 52	Assembles the updated program
IMPLICIT	DMSUPD	57	Update processing will be done using disk
INSEQW	DMSUPD	59	Processing when sequence errors occur in input file
INSLOOP	DMSUPD	59	Inserts cards from the source file
INVCHAR	DMSUPD	58	Processing for invalid character in sequence field
INVOPTN	DMSUPD	55	Error exit when an unrecognizable option is encountered
INVUPCD	DMSUPD	59	Processing for invalid update file control card
LOCTUPD	DMSUPD	54	Checks that a single update file exists
NOERASE	DMSUPD	54	Checks that the control file exists
NOFILE	DMSUPD	54	Processing when the source input file is not found
NOFILEW	DMSUPD	56	Processing when PTF file not found
NOFNAME	DMSUPD	55	Error exit when no operands were entered
NOUPDATS	DMSUPD	54	Abnormally terminates when update file specified but not found
OPTCONF	DMSUPD	55	Abnormally terminates when conflicting options specified
OPTDUP	DMSUPD	55	Abnormally terminates when the same option is specified more than once
PROCESS	DMSUPD	54	Checks if the update and source input files already exist
RETRD	DMSUPD	60	Creates disk output file from the in-storage updated file
RETR001	DMSUPD	60	Closes and renames the created output disk file
RETURN	DMSUPD		Checks RETCODE for indication of warning messages
RSEQDEF	DMSUPD	58	Sets the sequencing to 5 or 8 characters
RSEQERR	DMSUPD	58	Issues DMSUPD184W message
RSEQFIN	DMSUPD	58	Sets up for resequencing
SINGUPD	DMSUPD	54, 58	Applies a single update
SMALLCOR	DMSUPD	57	Insufficient storage to begin update
-STCTL	VMFASM	51	Checks for CNTRL file
-STSYS	VMFASM	51	Checks for the ASSEMBLE file
TEST	VMFDATE	53	Checks for the input file
UPDREAD	DMSUPD	58	Reads control cards
UPDSERR	DMSUPD	59	Issues DMSUPD186W message
VMFDATE	VMFDATE	53	Creates the UPDATES file
WOVF	DMSUPD	59	Issues DMSUPD176W message
WRETURN	DMSUPD	54	Issues DMSUPD177I message
XDELE	DMSUPD	58	Deletes line from storage
XWRITE	DMSUPD	59	Inserts line into storage
ZERSEQ	DMSUPD	58	Issues DMSUPD182W message

VMFLOAD Program Label Directory

Label	Module or Procedure	Figure	Description
BDCTR	VMFLOAD	61	Error exit when error occurs while reading control file
DINITA	VMFLOAD	61	Reads the MACS record from control file
DINITB	VMFLOAD	61	Punches text files
DINITD	VMFLOAD	61	Punches the highest level update available
ENDL	VMFLOAD	61	Closes punch and returns to CMS
FNDM	VMFLOAD	61	Searches for file specified in control file
NOCTR	VMFLOAD	61	Error exit when control file not found
NOFILE	VMFLOAD	61	Skips the files that are not found
NOLDL	VMFLOAD	61	Error exit when loadlist EXEC procedure is not found
RDCTR	VMFLOAD	61	Reads the control file
RETERR	VMFLOAD	61	Exits to CMS
SRTXT	VMFLOAD	61	Punches the TEXT file if update level is not found
VMFLOAD	VMFLOAD	61	Entry for load list program

VMFMAC Procedure Label Directory

Label	Module or Procedure	Figure	Description
-AREAD	VMFMAC	62	Checks that each macro or copy file listed in the 'maclibname EXEC' file exists
-ASGN	VMFMAC	62	Checks that the 'maclibname EXEC' file exists
-ERR2	VMFMAC	62	Prints error message if entire update procedure is not successful
-MACUP	VMFMAC	62	Updates the macro or copy files and puts them in the new macro library
-RENEWCO	VMFMAC	62	Renames existing NEWMAC COPY and NEWMAC MACLIB files
-STCTL	VMFMAC	62	Checks that the 'cntrlname CNTRL' file exists
-STKL	VMFMAC	62	Prints the control file
-UPDERR	VMFMAC	62	Prints error message if error occurs during updating

VMFTXT Procedure Label Directory

Label	Module or Procedure	Figure	Description
MAIN	VMFTXT	64	Checks that the invocation conditions are correct, that the 'txtlibname EXEC' file exists, and processes each entry in the 'txtlibname EXEC' file
WITH_TYPE	VMFTXT	64	Checks that those files listed in the 'txtlibname EXEC' file with a filetype do exist

Label	Module or Procedure	Figure	Description
FIND_TYPE	VMFTXT	64	Searches through the list of filetypes defined in the specified CNTRL file until a file with that filetype and given filename is found
ADD_TXT	VMFTXT	64	Adds the files to the VMFTXT TXTLIB
CNTRL	VMFTXT	64	Stacks the CNTRL file, reads the stack, verifies that only one MACS record exists and that it is the first record, and verifies that all entries have valid filetypes
VERFN	VMFTXT	64	Checks that the filename and filetype parameters are valid

Diagnostic Aids

The following figures list all the messages issued by the modules and EXEC procedures that create and update the VM/SP system Figure 54 on page 126 lists all the messages issued by the VMFASM EXEC procedure, Figure 55 on page 127 lists the messages issued by the DMSUPD module, Figure 56 on page 128 lists the messages issued by the VMFLOAD program, and Figure 57 on page 129 lists the messages issued by the VMFMAC procedure. The label of the issuing routine and the figure (if any) describing that routine are included.

VMFASM Procedure Messages

Label	Figure	Message Text
-FUPD	52	***ERROR UPDATING filename***
-ASMP	52	ASMBLING filename (options...)
-DTF	52	***ERROR ASMBLING filename***
-DTF	52	***NO TEXT FOR filename***
-COMB	52	TEXT filename TXTxxxxx CREATED

DMSUPD Program Messages

Message Code	Label	Figure	Return Code or Severity	Message Text
DMSUPD001E	NOFNAME	55	24	No filename specified
DMSUPD002E	NOFILE	54	28	File [*fn [ft [fm]]*] not found
DMSUPD003E	INVOPTN	55	24	Invalid option: *option*
DMSUPDOO7E	FMTERR		32	File *fn ft fm* is not fixed, 80-character records
DMSUPD010W	INPFERR		12	Premature EOF on file *fn ft fm*--sequence number *seqno* not found
DMSUPD024E	PROCESS ERCMSUT	54	24	File *fn ft fm* already exists
DMSUPD037E	PROCESS ERRW	54	36	Disk *mode* (*vdev*) is accessed as read/only
DMSUPD048E	BADMODE		24	Invalid mode *mode*
DMSUPD065E	OPTDUP	55	24	*option* option specified twice
DMSUPD066E	OPTCONF	55	24	*option1* and *option2* are conflicting options
DMSUPD069E	NOTACCER		32	Disk *mode* not accessed
DMSUPD070E	EXCESIV	55	24	Invalid parameter *parameter*
DMSUPD104S	INPERR		100	Error *nn* reading file *fn ft fm* from disk

Message Code	Label	Figure	Return Code or Severity	Message Text
DMSUPD105S	OUTERR		100	Error *nn* writing file *fn ft fm* on disk
DMSUPD174W	ZINSLOOP ZPASSW	59	8	Sequence error introduced in output file: *seqno1* to *seqno2*
DMSUPD176W	WOVF	59	8	Sequencing overflow following sequence number *seqno*
DMSUPD177I	WRETURN	54	-	Warning messages issued (severity = nn)[; REP option ignored]
DMSUPD178I	CTLUMSG	56	-	Updating *fn ft fm* Applying *fn ft fm*
DMSUPD179E	ERMACS	56	32	Missing or duplicate MACS card in control file *fn ft fm*
DMSUPD180W	NOFILEW	56	12	Missing PTF file *fn ft fm*
DMSUPD181E	NOUPDATS	54	40	No update files were found
DMSUPD182W	ZERSEQ	58	8	Sequence increment is zero
DMSUPD183E	BADCTLC BADAUXC	56	32	Invalid {CONTROL|AUX} file control card
DMSUPD184W	RSEQERR	58	12	./S not first card in update file--ignored
DMSUPD185W	INVCHAR	58	12	Invalid character in sequence field *seqno*
DMSUPD186W	UPDSERR	59	12	Sequence number *seqno* not found
DMSUPD187E	ERSC	55	24	Option STK invalid without CTL
DMSUPD207W	UPDREAD INVUPCD	58 59	12	Invalid update file control card
DMSUPD210W	INSEQW	59	4	Input file sequence error: *seqno1* to *seqno2*
DMSUPD299E	CORBUST	59	40	Insufficient storage to complete update
DMSUPD300E	SMALLCOR	56	40	Insufficient storage to begin update
DMSUPD304I	IMPLICIT	56	-	Update processing will be done using disk
DMSUPD361E				Disk *mode* is not a CMS disk

VMFLOAD Program Messages

Label	Figure	Message Text
NOFILE	61	filename filetype NOT FOUND
BDCTR	61	ERROR IN CONTROL FILE
NOCTR	61	NO CONTROL FILE
NOLDL	61	NO LOAD LIST
ENDL	61	SYSTEM LOAD DECK COMPLETE

VMFMAC Procedure Messages

Label	Figure	Message Text
-ASGN	62	*** maclibname EXEC NOT FOUND ***
-STCTL	62	*** cntrlname CNTRL NOT FOUND ***
-AREAD	62	*** filename COPY OR MACRO NOT FOUND ***
-UPDERR	62	*** ERRORS UPDATING membername membertype *** membername membertype NOT INCLUDED IN MACLIB
-ERR2	62	DUE TO PREVIOUS ERRORS, THE RESULTS OF THIS MACLIB BUILD IS CALLED 'NEWMAC MACLIB', libname MACLIB HAS NOT BEEN REPLACED

VMFTXT Procedure Messages

Message Code	Label	Figure	Message Text
DMSWTX001E	MAIN	64	No filename specified
DMSWTX002E	MAIN	64	File *fn ft fm* not found
DMSWTX006E	MAIN	64	No read/write A disk accessed
DMSWTX024E	MAIN	64	File *fn ft fm* already exists
DMSWTX026E	MAIN	64	Invalid parameter *parameter* for *function* function
DMSWTX056E	MAIN	64	File *fn ft [fm]* contains invalid record formats
DMSWTX062E	MAIN	64	Invalid character *char* in fileid *fn ft*
DMSWTX179E	CNTRL	64	Missing or duplicate MACS card in control file *fn ft fm*
DMSWTX183E	CNTRL	64	Invalid control file card
DMSWTX895I	MAIN	64	Member *fn ft* added
DMSWTX895I	ADD_TXT	64	Member *fn ft* added
DMSWTX896E	WITH_TYPE	64	File *fn ft fm* not found
DMSWTX896E	FIND_TYPE	64	File *fn* TEXT or *fn* TXT* not found
DMSWTX897E	MAIN	64	Due to previous errors, the result of this TXTLIB build is called VMFTXT TXTLIB; your *fn* TXTLIB has not been replaced.

VMFNLS EXEC Messages

Message Code	Label	Figure	Return Code or Severity	Message Text
DMSWNL001E	FATAL		24	No filename specified
DMSWNL002E	FATAL		24	File *fn ft* not found
DMSWNL023E	FATAL		24	No filetype specified
DMSWNL032E	FATAL		24	Invalid filetype *ft*
DMSWNL122E	FATAL		*rc*	Return code *rc* from *command*
DMSWNL328E	FATAL		24	Control file not specified
DMSWNL448E	FATAL		28	Country code *code* not in VMFNLS LANGLIST

DMKLD00E (Loader) Program

If the loader terminates, one of the following wait conditions is indicated in the instruction counter:

Code	Meaning
X'111111'	A program check occurred.
X'222222'	A unit check occurred while the bootstrap routine was reading in the loader.
X'999999'	An SVC was issued.
X'BBBBBB'	A machine check occurred.
X'CCCCCC'	An I/O error occurred on the card reader.
X'FFFFFF'	An I/O error occurred for the console (X'00' contains the message UNRECOVERABLE ERROR), or the control card for changing the default I/O addresses for the printer or terminal is invalid (X'00' contains the message BAD DEVICE CARD or INVALID DEVICE SPECIFIED).

Loader Wait State Codes

If the instruction counter contains X'999999', indicating an SVC wait state, examine the interruption code (the third and fourth bytes of the supervisor old PSW). The interruption codes (shown in hexadecimal) have the following meanings:

Code	Meaning
64	An error occurred during conversion of a value from hexadecimal to binary format.
65	There is no more free storage available for the loader.
66	A duplicate type 1 ESD (External Symbol Dictionary) entry has been encountered.
67	The "name" in the LDT (Loader Terminate) statement is undefined.
68	The control section named in the ICS (Include Control Section) statement was not found by end of file.
69	The loader attempted to add another entry to the reference table, which would have caused the table to overflow.
6A	The object modules being loaded are about to overlay the loader.
6B	The object modules being loaded are about to overlay an address between zero and 100.
6C	A permanent error occurred in the input device.
6D	The loader is trying to release storage that is not on a doubleword boundary.

For further explanations of these wait state conditions and the recommended operator action to correct them, see *VM/SP System Messages and Codes*.

Index

W

Chapter 6. VM/SP Starter System

VM/SP Starter System

Introduction

The Starter System Program (DMKSSP) redefines the real configuration according to the operator's specifications.

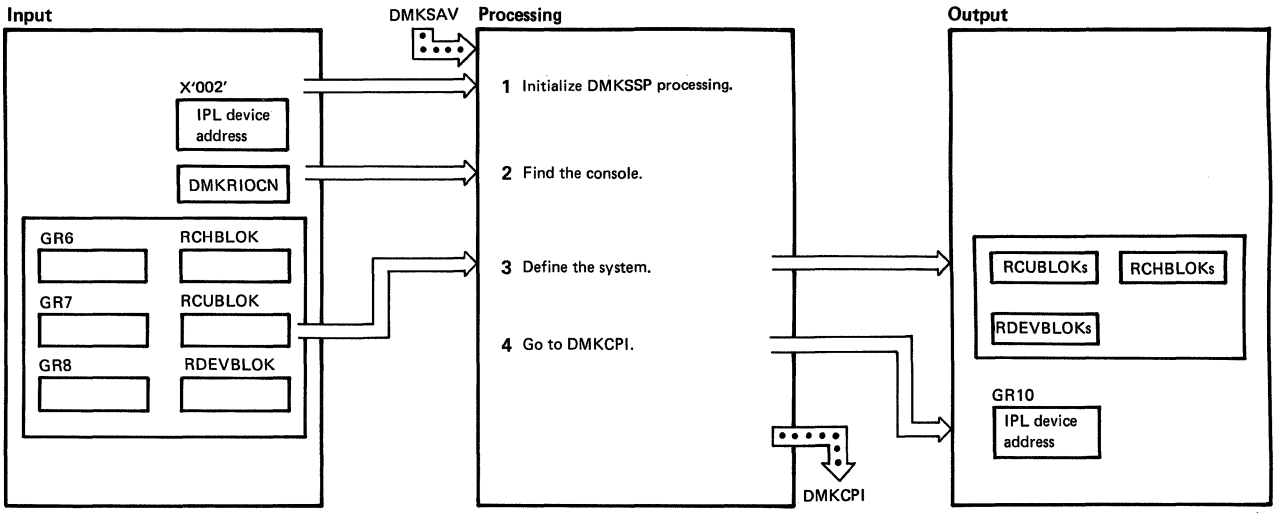
Normally, VM/SP is loaded from disk. The DMKSAV module reads a copy of the CP nucleus into real storage and then calls DMKCPI to perform the initialization tasks (such as initializing storage, mounting devices, and so on). However, during system generation, the VM/SP starter system is loaded from the starter system tape to disk using DDR. Then when VM/SP is loaded, the DMKSAV module reads a copy of the starter system nucleus into real storage and calls DMKSSP to give the operator the opportunity to redefine the devices necessary to continue with system generation. When DMKSSP is through with its processing, it calls DMKCPI to continue the initialization process.

DMKSSP is an interactive program. The operator must signal attention to define a console at an address other than 009 or 01F. Then, the operator responds to questions displayed at the terminal to redefine the printer, punch, reader, tape and disk devices.

Method of Operation

This section describes those functions that are performed by the DMKSSP program.

VM/SP Starter System



Notes		Module	Label	Ref
1	Registers 11 and 12 are set up as base registers. The new I/O PSW, new machine check PSW, and new program check PSW are set up and all interrupts are disables.	DMKSSP	DMKSSP01	
2	If the console address is valid, DMKSSP displays VM/SP STARTER SYSTEM ***DO YOU WISH TO REDEFINE YOUR SYSTEM*** (YES,NO): If the response is YES, proceed by redefining the system (see step 3). If the response is NO, DMKSSP processing is done. Proceed to step 4.	DMKSSP	HDRMSG REDEFINE	
3	First, all control blocks and their pointers are cleared and the system residence device is set up.	DMKSSP	MAINLINE	
	DMKSSP must find the console. If the console is not at 009 or 01F, DMKSSP enables for interrupts and waits until the operator signals attention to identify the console. The CPU model is checked and if it is valid, DMKSSP builds the real control blocks for the console, and displays VM/SP STARTER SYSTEM		FINDCONS VLDCOND	
	DMKSSP prompts the operator to reconfigure the system, DMKSSP displays ENTER PRINTER ADDRESS (cuu): ENTER DEVICE TYPE (1403, 1443, 3211, 3203, 3262, 3289E, 4245, 4248): and builds the printer real control blocks according to the operator's response.		HDRMSG PRTLAB	
	DMKSSP displays ENTER PUNCH ADDRESS (cuu): ENTER DEVICE TYPE (2540P, 3525): and builds the punch real control blocks according to the operator's response.		PCHLAB	
	DMKSSP displays ENTER READER ADDRESS (cuu): ENTER DEVICE TYPE (2540R, 2501, 3505): and builds the reader real control blocks according to the operator's response.		RDRLAB	
	DMKSSP displays ENTER ADDRESS WHERE FIRST TAPE IS MOUNTED (cuu): ENTER DEVICE TYPE (3420, 2415, 2420, 2401, 3430, 8809): and builds the tape real control blocks according to the operator's response.		PIDLAB	
	DMKSSP displays ENTER ADDRESS OF A SECOND TAPE DIRVE (cuu): ENTER DEVICE TYPE (3420, 2415, 2420, 2401, 3430, 8809): and builds the tape real control blocks according to the operator's response.		BKUPLAB	
	DMKSSP displays ENTER DEVICE ADDRESS OF WORK PACK (cuu): ENTER DEVICE TYPE (3330, 3340, 3350, 3375, 3380, FB-512): and builds the disk real control blocks according to the operator's response.		GRAFED	
	DMKSSP displays ENTER ADDRESS WHERE EXTRA WORK PACK IS MOUNTED (cuu): ENTER DEVICE TYPE (3330, 3340, 3350, 3375, 3380, FB-512):		SYSLAB	
	DMKSSP displays ENTER ADDRESS OF A GRAPHIC DEVICE (cuu): ENTER DEVICE TYPE (3277, 3278, 3066)		WORKLAB	

Figure 65 (Part 1 of 2). DMKSSP--The Starter System

Notes		Module	Label	Ref
	DMKSSP then asks the operator to verify the configuration by displaying ***SYSTEM DEFINITION COMPLETED*** cuu PRINTER cuu PUNCH cuu READER cuu FIRST TAPE cuu SECOND TAPE cuu WORK PACK cuu EXTRA WORK PACK cuu GRAPHIC DEVICE ARE THE ABOVE ENTRIES CORRECT (YES,NO):	DMKSSP	NOWORK	
	If the operator responds NO, the entire system definition process is repeated.		XPRINT	
4	Control is transferred to DMKCPI with the address of the IPL device in general register 10.			

Figure 65 (Part 2 of 2). DMKSSP--The Starter System

Program Organization

This section describes the organization of the DMKSSP module.

DMKSSP

The Starter System Program that allows the operator to redefine the minimum devices necessary to generate the VM/SP system.

Attributes

Nonreentrant, resident, entered via IPL.

Entry Conditions

DMKSSP001 is entered as the result of an IPL.

Exit Conditions

DMKSSP gives control to DMKCPINT to initialize the remainder of the system. Register 10 must contain the IPL device address.

Register Use

Reg	Use
R1	Parameter register
R2	Parameter register
R5	General BAL register
R6	Address of RCHBLOK
R7	Address of RCUBLOK
R8	Address of RDEVBLOK
R11	Base register 2
R12	Base register 1

External References

Routine	Purpose
DMKRIODV	Anchor to the first real device block
DMKRIOCU	Anchor to the first real control unit block
DMKRIOCH	Anchor to the first real channel block
DMKRIOCN	Address of the system console device
DMKRIOPR	Address of the system printer device
DMKRIOPU	Address of the system punch device
DMKRIORD	Address of the system reader device
DMKSYSNU	Disk address on the nucleus
AMKRIO	Address of real I/O control blocks

Call to Other Routines

Routine	Purpose
DMKCVTHB	To convert the device address to binary
DMKCVTBH	To convert the device address to printable hexadecimal characters
DMKCPINT	To continue system initialization

Data Areas

RCHBLOK, RCUBLOK, RDEVBLOK, PSA

Directory

Following is an alphabetic list of the major labels in the Starter System Program. The associated method of operation diagram (if any) is indicated and a brief description of the operation performed at the point in the program associated with each label is included.

Label	Figure	Description
ATTNHAND	65	Enables system for I/O interrupts
BKUPLAB	65	Builds real control blocks for scratch tape
DASDADR	65	Sets up device type for disk containing the starter system
DMKSSP01	65	Starter system entry point called by DMKSAV
FINDCONS	65	Identifies the system console
GRAPHID	65	Handles the I/O for display terminals
HDRMSG	65	Displays starter system header message
MAINLINE	65	Builds all the real control blocks necessary
PCHLAB	65	Builds the real control blocks for the punch
PIDLAB	65	Builds the real control blocks for the tape drive containing the PID (Program Information Department) distribution tape
PRTLAB	65	Builds the real control blocks for the printer
RDRLAB	65	Builds the real control blocks for the reader
READADDR	65	Initiates writes to and reads from the console to determine the device address
READTYPE	65	Initiates writes to and reads from the console to determine the device type
REAWRITE	65	Writes to and reads from the console. The REAWRITE routine is called by both the READADDR and READTYPE routines.
REDEFINE	65	Asks the operator if he wants to redefine the system
SCAN	65	Finds or builds the necessary real control blocks
STARTIO	65	Issues the Start I/O (SIO)
SYSLAB	65	Builds the real control blocks for the disk that contains the system residence volume
VLDCON	65	Checks for a valid CPU model
WORKLAB	65	Asks the operator if the configuration just defined is the one he wants
XFRINIT	65	Transfers control to DMKCPI

Diagnostic Aids

Following is a list of the messages issued by the Starter System Program. The associated program label and method of operation diagram are included in the list.

Label	Figure	Message Text
BKUPLAB	65	ENTER ADDRESS WHERE SCRATCH TAPE IS MOUNTED (cuu): ENTER DEVICE TYPE (2401, 2415, 2420, 3420, 3430, 8809):
HDRMSG	65	VM/SP STARTER SYSTEM VERSION n.n
PCHLAB	65	ENTER PUNCH ADDRESS (cuu): ENTER DEVICE TYPE (2540P, 3525):
PIDLAB	65	ENTER ADDRESS WHERE PID TAPE IS MOUNTED (cuu): ENTER DEVICE TYPE (2401, 2415, 2420, 3420, 3430, 8809):
PRTLAB	65	ENTER PRINTER ADDRESS (cuu): ENTER DEVICE TYPE (1403, 1443, 3203, 3211, 3262, 3289E, 3800):
RDRLAB	65	ENTER READER ADDRESS (cuu): ENTER DEVICE TYPE (2501, 2540R, 3505):
REDEFINE	65	***DO YOU WISH TO RE-DEFINE YOUR SYSTEM*** (YES,NO):
SYSLAB	65	ENTER DEVICE ADDRESS WHERE SYSTEM RESIDENCE WILL BE BUILT (cuu): ENTER DEVICE TYPE (2319, 3330, 3340, 3350, 2305, FB-512, 3380):
WORKLAB	65	***SYSTEM DEFINITION COMPLETED*** cuu PRINTER cuu PUNCH cuu READER cuu PID TAPE cuu SCRATCH TAPE cuu NEW SYSTEM RESIDENCE cuu SCRATCH PACK ARE THE ABOVE ENTRIES CORRECT (YES,NO):
WNGDEV		**ERROR** DEVICE HAS BEEN ALREADY ALLOCATED

Index

Chapter 7. 3704/3705 Service Programs

Introduction

Note: To generate an IBM 3725 control program, refer to the *EP Generation and Utilities Guide,* SC30-3172.

There are four CMS commands and two CP commands specifically for generating and manipulating the 3704/3705 control program. The CMS commands are needed to generate and save a copy of the 3704/3705 control program. The CP commands allow you to operate and manipulate the 3704/3705 in a manner similar to the way other CP commands let you operate your other virtual machine devices.

The CMS commands that help you generate a 3704/3705 control program are: ASM3705, GEN3705, LKED, and SAVENCP. The ASM3705 command is an interface between CMS and the NCP/VS Release 2 and 3 Assembler (IFKASM) or the NCP/VS Release 4 Assembler (CWAX00). It accepts source statement files as input, checks that the input file exists and that the options specified are valid, calls IFKASM or CWAX00 to perform the assembly, and produces an object deck and program listing as output. The ASM3705 command produces the stage 1 output for the 3704/3705 control program generation process.

The GEN3705 command accepts the file produced in stage 1, creates a unique assembler file for each job step in the input file, creates several unique files containing the linkage editor statements necessary to build the load module file, and builds an EXEC macro file of the CMS commands necessary to assemble and load the 3704/3705 control program. If SAVE was specified on the command line, it saves a copy of the control program in page-format on a CP-owned volume.

The LKED command is an interface between CMS and the OS/VS1 linkage editor. The GEN3705 command processor embeds the LKED commands in the EXEC macro file it produces. The LKED command processor interprets the CMS command lines, defines the necessary files, and links to the OS/VS linkage editor. For information about the LKED command, see the *VM/SP CMS Command Reference.*

The SAVENCP command builds the parameter list (CCPARM) and calls DMKSNC via Diagnose instruction X'50' to write a core image copy of the 3704/3705 control program to a CP-owned system volume. This copy of the control program is loaded each time the 3704/3705 is loaded.

The CP commands that help you to control the operation of the 3704/3705 are NCPDUMP and NETWORK. The NCPDUMP command processor performs several different tasks. It does the following:

- Erases a specific CP or CMS 3704/3705 dump file
- Formats the 3704/3705 dump

- Prints the 3704/3705 dump file
- Assigns an identifier to the 3704/3705 dump file
- Creates the CMS 3704/3705 dump file.

The NETWORK command processor provides the support for the 3704/3705 that several CP commands (ENABLE, DISABLE, QUERY, DISPLAY, VARY, HALT, TRACE, and SHUTDOWN) provide for other devices. In addition, the NETWORK command has options that load a named 3704/3705 control program into 3704/3705 storage and dump the contents of that storage.

These commands are discussed in detail in other publications. For more information about the ASM3705, GEN3705, and SAVENCP commands and a complete description of the generation process, see the VM/SP *Planning Guide and Reference* and *Installation Guide.* For more information about the NCPDUMP and NETWORK commands, see the VM *Diagnosis Guide.*

The ZAP service program, which allows you to update and dump existing 3704/3705 load libraries, is described in "The ZAP Service Program" and in the VM/SP *Operator's Guide.*

Method of Operation

This section describes the CMS modules that provide the commands to generate the 3704/3705 control programs. Diagrams describe the functions performed by each of the command processors. Figure 66 shows the relationships between these diagrams.

Figure 67 on page 167 describes the SAVENCP command, which saves an image of the 3704/3705 control program so that it can later be loaded. Figure 68 on page 168 shows how CCPARM is built.

Figures 69, 70, and 71 describe the GEN3705 command, which generates a series of commands to assemble, link edit, and load the 3704/3705 control program.

Figures 72 and 73 describe the ASM3705 command, which is an interface between CMS and the NCP/VS Assembler (IFKASM or CWAX00).

Figure 74 on page 175 describes the NCPDUMP command, which prints a dump of the 3704/3705 storage.

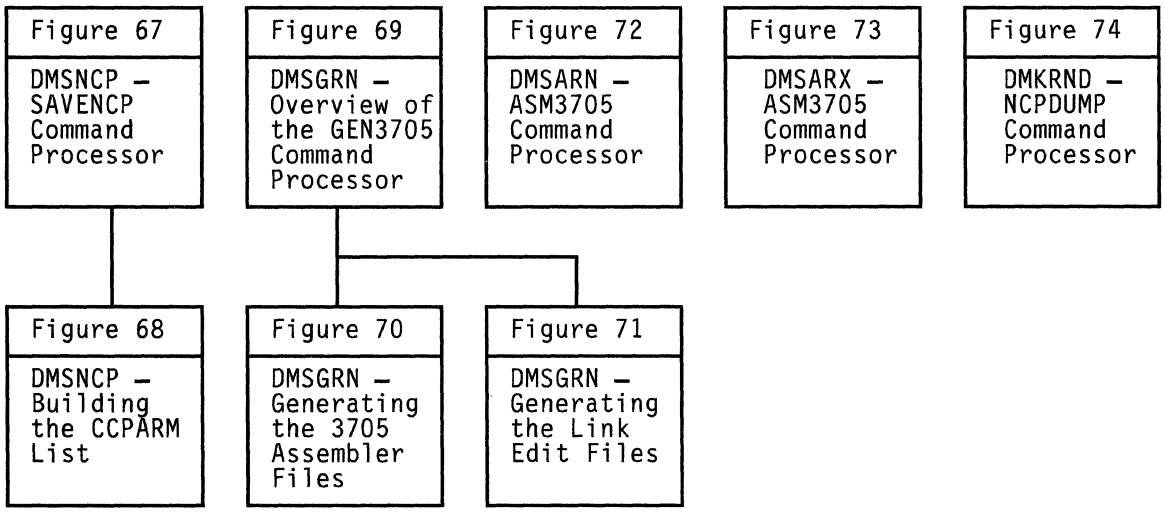
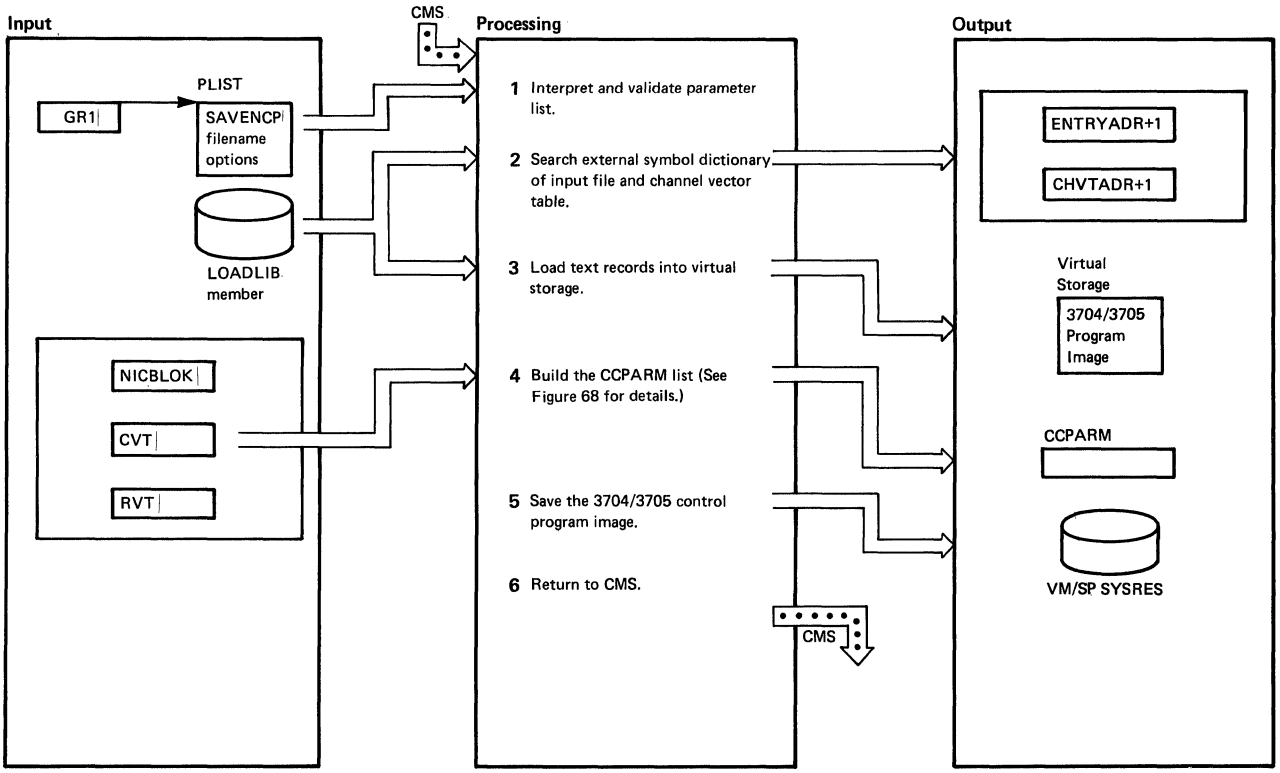
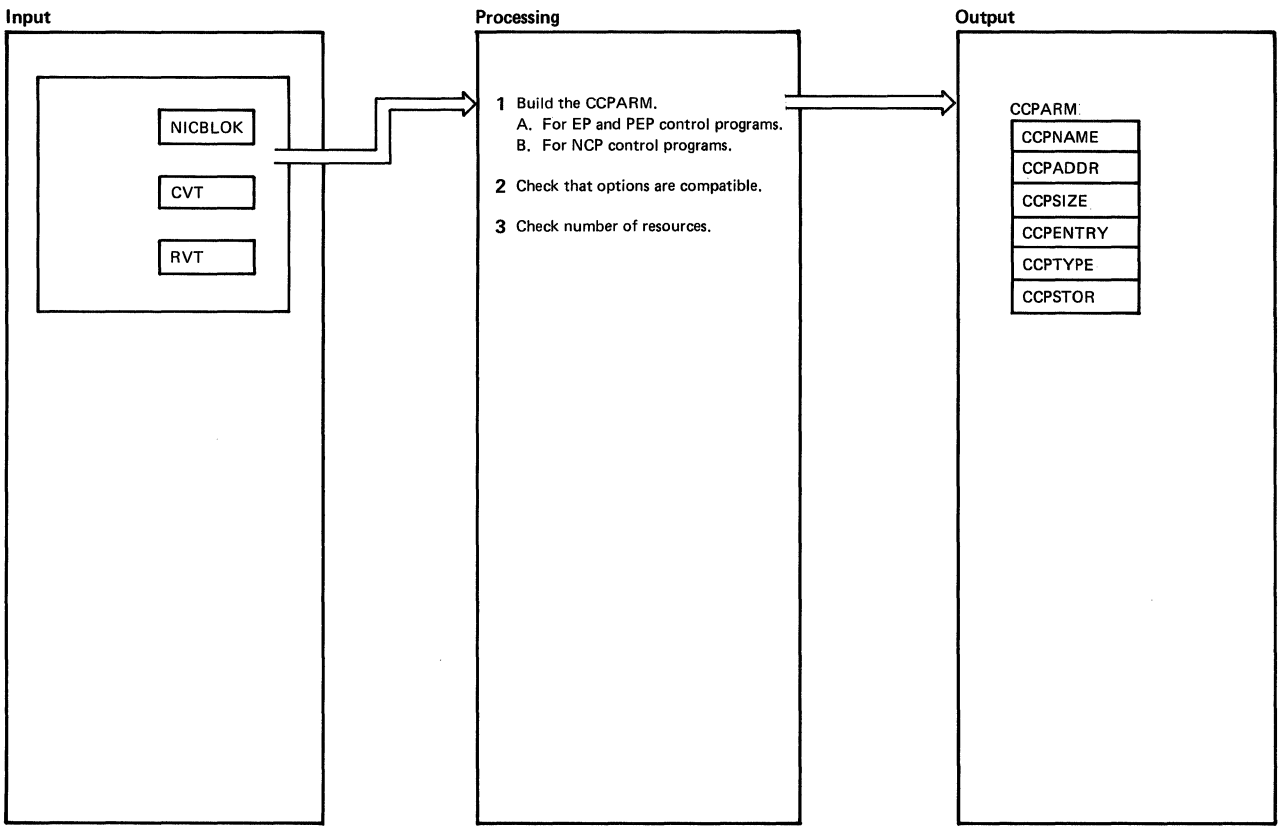


Figure 66. Key to the 3704/3705 Service Programs Method of Operation Figures



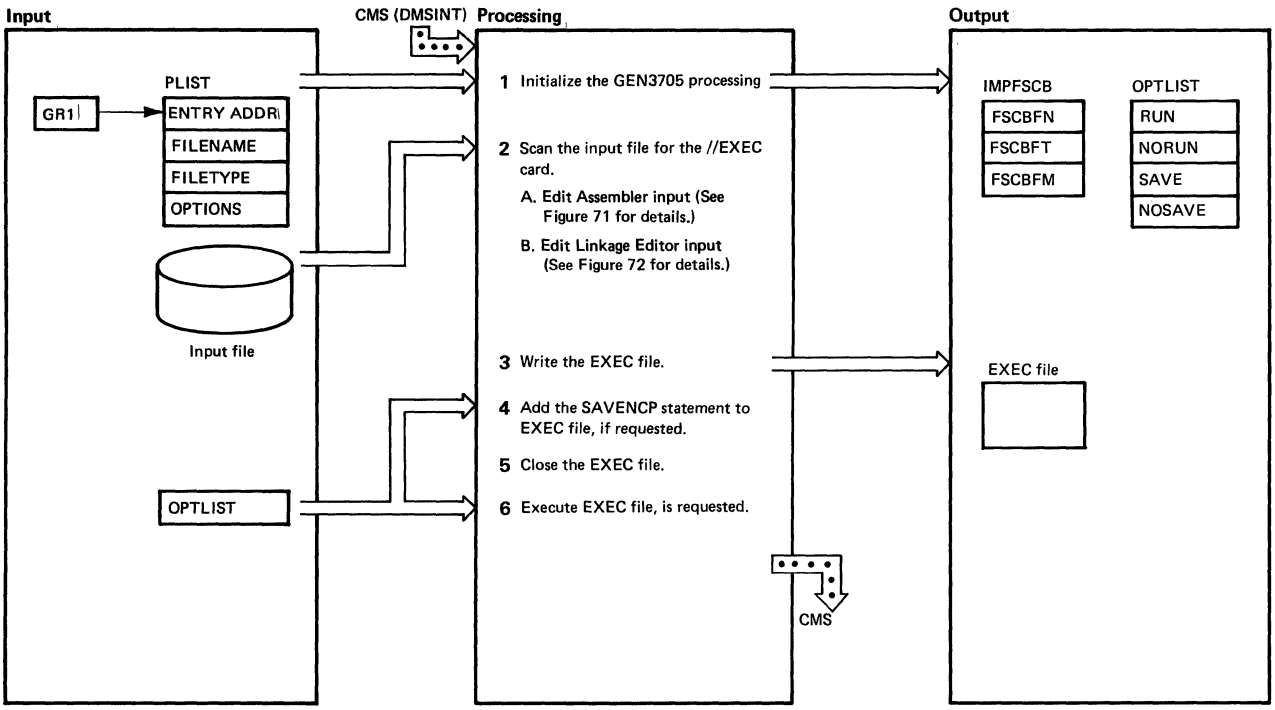
Notes		Module	Label	Ref
1	The filename must be specified. If a library name or a member name is not specified, the input filename is used. If the 3704/3705 control program load module entry point is not specified. CXFINIT is assumed. An error in the parameter list results in one of the following messages DMSNCP001E NO FILENAME SPECIFIED DMSNCP002E File *fn [[fm]]* not found DMSNCP003E Invalid option: *option* being issued and control being returned to CMS with return code 24 or 28. If no errors are encountered, the input file is opened and a search is made for the member. When the member is found, it is read. If the member is not found, the message DMSNCP013E MEMBER xxxxxxxx NOT FOUND IN LIBRARY is issued and control returns to CMS with a return code of 4.	DMSNCP	SAVENCP ENDPARMS DOSTATE	
2	The entry point for NCP or PEP is CXFINIT. The entry point for EP is CYASTART. For either EP or PEP, the channel vector table, CYACHVT, CYECHVT1, or CYECHVT2 must also be found. The entry point address and channel vector table address are saved.	DMSNCP	CESDENT CESDCHVT	
3	The text records are moved from the input buffer into the proper position in the core image buffer. If the entry point symbol has not been resolved when the first text record is encountered, the message DMSNCP021E ENTRY POINT xxxxxxxx NOT FOUND is issued and control returns to CMS with a return code of 40. Premature end of file or invalid control records cause the messages DMSNCP056E FILE 'fn ft' CONTAINS INVALID RECORD FORMATS DMSNCP109E VIRTUAL STORAGE CAPACITY EXCEEDED to be issued and control to be returned to CMS.	DMSNCP	CONTROL ERR21 ERR66	
4	When the core image buffer is loaded, the input file is closed. The Communication Control Parameter list (CCPARM) is built from the information in the core image buffer.	DMSNCP	CLOSE	
5	The size of the read buffer is stored in register 1 and the DIAGNOSE instruction with code X'50' is issued to save a copy of the 3704/3705 control program.	DMSNCP	SAVECCP	
6	The return code from the DIAGNOSE instruction is passed to CMS and control returns to CMS.	DMSNCP	EXIT	

Figure 67. DMSNCP--SAVENCP Command Processor



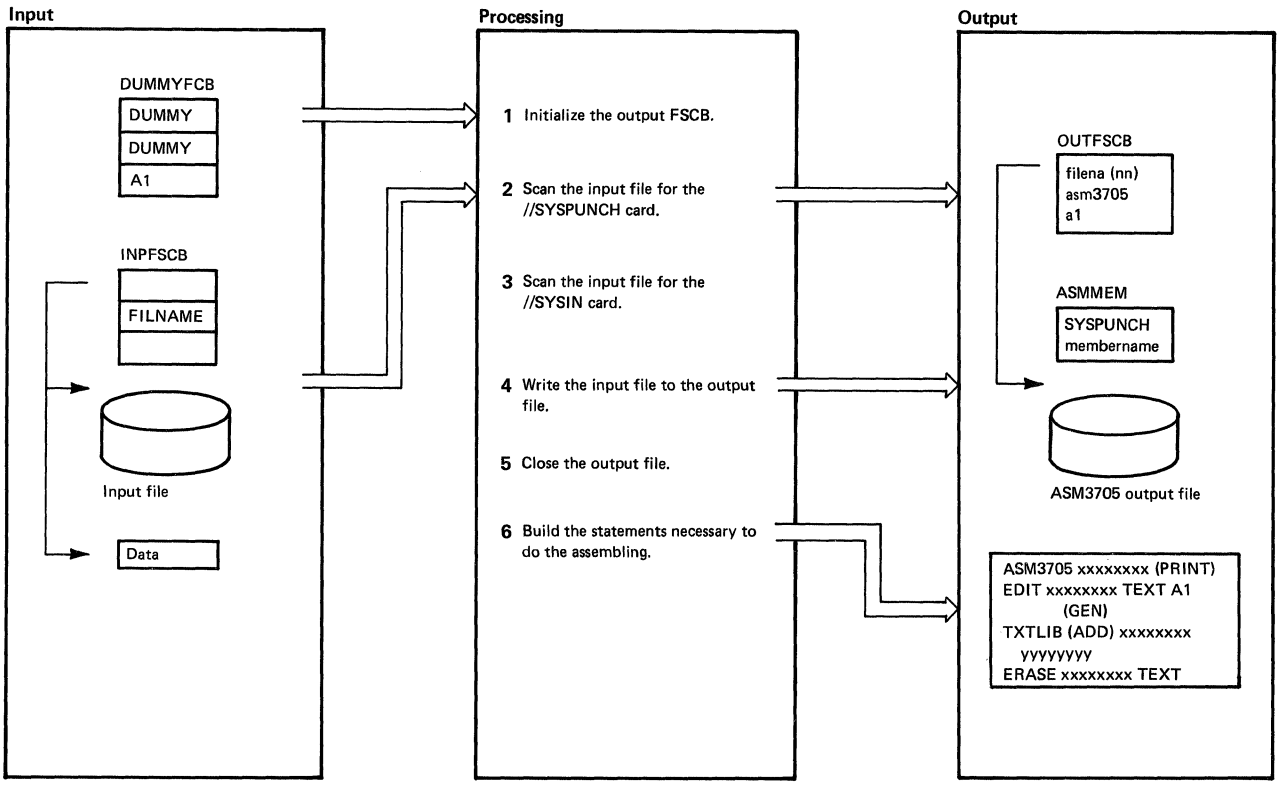
Notes	Module	Label	Ref
1	DMSNCP		
A. For EP and PEP control programs, additional fields are updated (CCPRSTYP, CCPRSTAT, CCPRSTEP, CCPPSIZE). A channel vector table must exist for EP and PEP control programs. If the CVT does not exist, the message DMSNCP025E INVALID DATA IN 370X PROGRAM is issued and control returns to CMS with return code 16.		SCANCEP	
B. Additional fields in the CCPARM block are updated for NCP and PEP contorl programs (CCPCAONE, CCPHBFSZ, CCPHBFNO, CCPPADO, CCPPADI, CCPMAXID, CCPRESID, CCPRSTYP, CCPRSTAT, CCPRSTEP).		SCANNCP	
2 A check is made that the options specified are compatible. If they are not, the message DMSNCP099W GENERATION PARAMETERS INCOMPATIBLE WITH VM/SP is issued and processing continues.	DMSNCP	CHEKVMV	
3 If there are more than 4086 resources or if the first resource is not a 3704/3705, the message DMSNCP025E INVALID DATA IN 370X PROGRAM is issued and control returns to CMS with a return code of 16.	DMSNCP		

Figure 68. DMSNCP--Building the CCPARM List



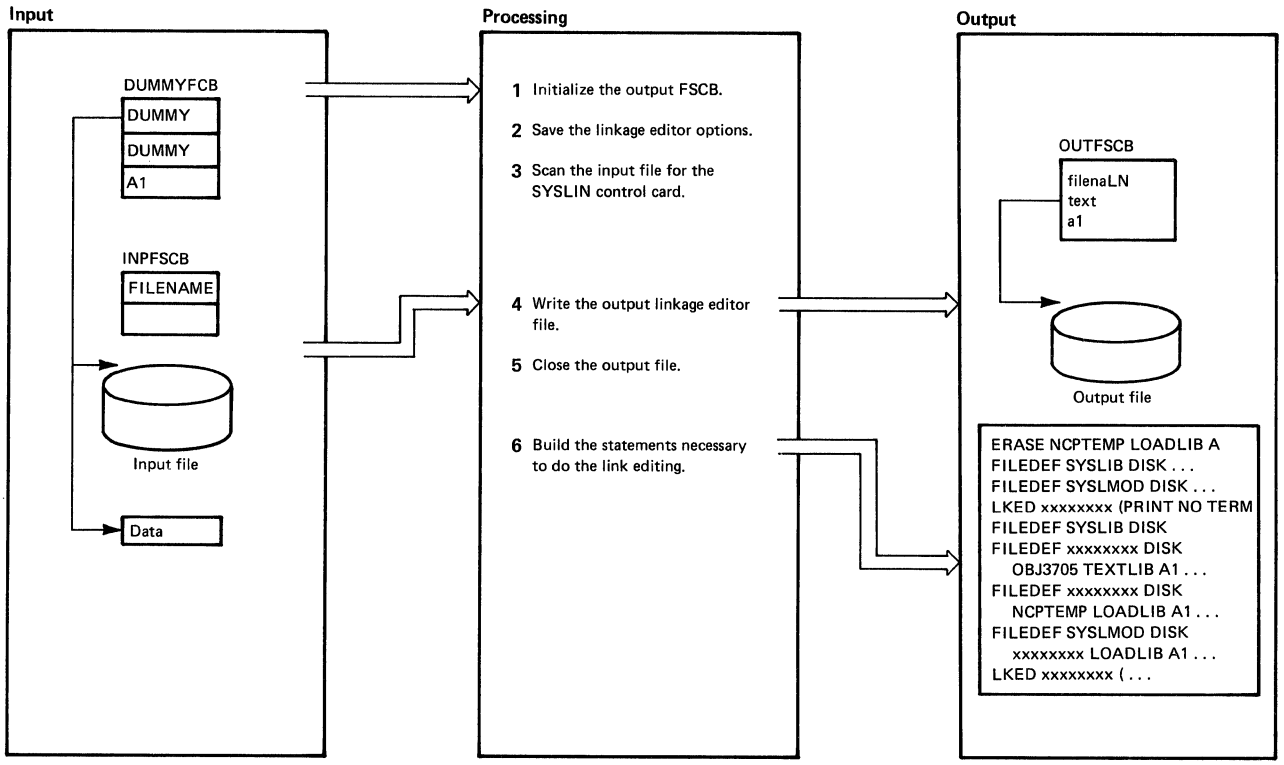
Notes		Module	Label	Ref
1	The input file name, type, and optionally the mode are put into INPFSCB. The filename or the first 6 characters of the name, whichever is the least, is saved for naming the assembler and linkage editor output files.	DMSGRN	START	
	The input options are scanned and the appropriate options are set on. Invalid options cause the message DMSGRN003E Invalid option: *option* to be issued.		OPTIONS1	
	The FSSTATE macro is issued to see if the file exists. Either of the following messages is issued in case of an error DMSGRN048E Invalid Mode *mode* DMSGRN002E File *fn [ft [fm]]* not found		OPTEND	
2	The FSCBRRD routine is used to read the input file. The EDITIN routine scans for a //EXEC card containing PGM=IFKASM or PGM=IEWL. Control cards are scanned until a valid EXEC card is found. If *,//, or /* do not appear as the first characters of the input record or if an invalid //EXEC card is read, the message DMSGRN078E Invalid card in input file *fn* is displayed.	DMSGRN	PRIMEDIT	
	The IFKASM routine processes the assembler input and the IEWL routine processes the linkage editor input. After the input is processed, DMSGRN continues by scanning the input file for another //EXEC card.		FINDASM FINDIEWL	
3	The EXEC statements that were generated as a result of the assembler and linkage editor input are written to an EXEC file.	DMSGRN	STACK30	
4	The CLOSTACK routine is called to add SAVECP filename (ENTRY entryname to the end of the EXEC file, if SAVE was specified on the GEN3705 command.	DMSGRN	PROCEND2 STACK30	
5	The EXEC macro file is closed by branching and linking to the PROCEND routine.	DMSGRN	PROCEND1	
6	If RUN was specified, the command EXEC ncpname is stacked in the reader.	DMSGRN	PROCEND1	
	Control is returned to CMS.		RETURN1	

Figure 69. DMSGRN--Overview of the GEN3705 Command Processor



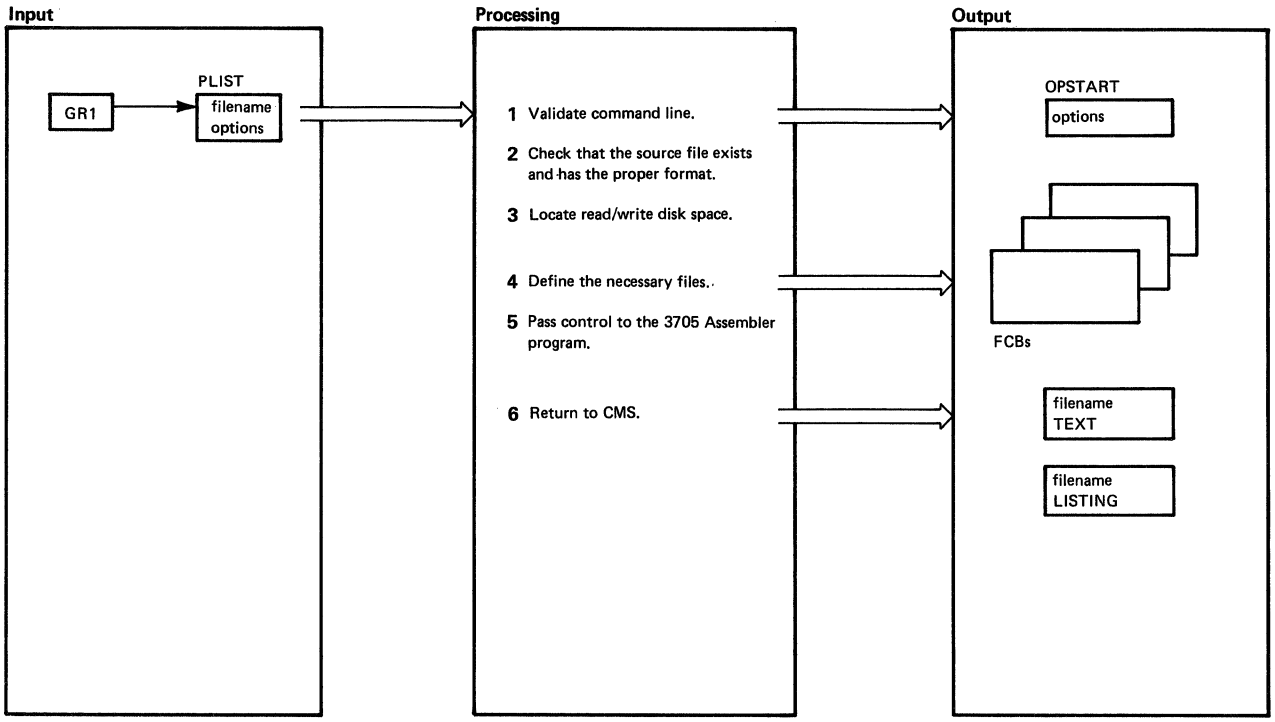
Notes		Module	Label	Ref
1	The filetype in the dummy FSCB is initialized to ASM3705. Each ASM3705 file has a filename consisting of the first 6 characters of the filename (or the entire filename if it is 6 characters or less) concatenated with a number. The FSCBWT routine uses the dummy FSCB to initialize the OUTFSCB.	DMSGRN	IFKASM	
2	The input file is scanned for a SYSPUNCH or SYSPUNCH continuation card. If found, it is scanned for the DSN= or DSNAME=keyboard. The DSNEDIT routine then saves the membername of the data set in the current SYSPUNCH membername savearea.	DMSGRN	IFKASM10 IFKASM34	
3	The input file is scanned for the SYSIN card. All cards scanned preceding the SYSIN card must have * or // in the first positions of the card. Otherwise DMSGRN078E Invalid card in input file *fn* is issued.	DMSGRN	IFKASM40	
4	The FSCBRD routine reads all the input and the FSCBWT routine writes it to the output file.	DMSGRN	IFKASMA0	
5	The output file is closed by branching and linking to the FSCBCLOS routine. Close errors are ignored.	DMSGRN	IFKASMK0	
6	The ASMFIRST bit in the PROCSW1 byte is tested. If the bit is on, the GEN parameter in the TXTLIB command is changed to ADD. Otherwise the bit is turned on.	DMSGRN	ASMSTAK	
	The name of the output assembler file is moved into the ASM3705 and EDIT commands. The FSCB base address is changed and the name of the input file is put into the TXTLIB command.		ASMSTAK2	
	The SYSPUNCH membername is then moved to the TXTLIB command.		ASMSTAK4	
	The number of commands and the address of the first command in the stack are loaded from STACKASM into registers 1 and 2 respectively.		ASMSTAK6	

Figure 70. DMSGRN--Generating the 3705 Assembler Files



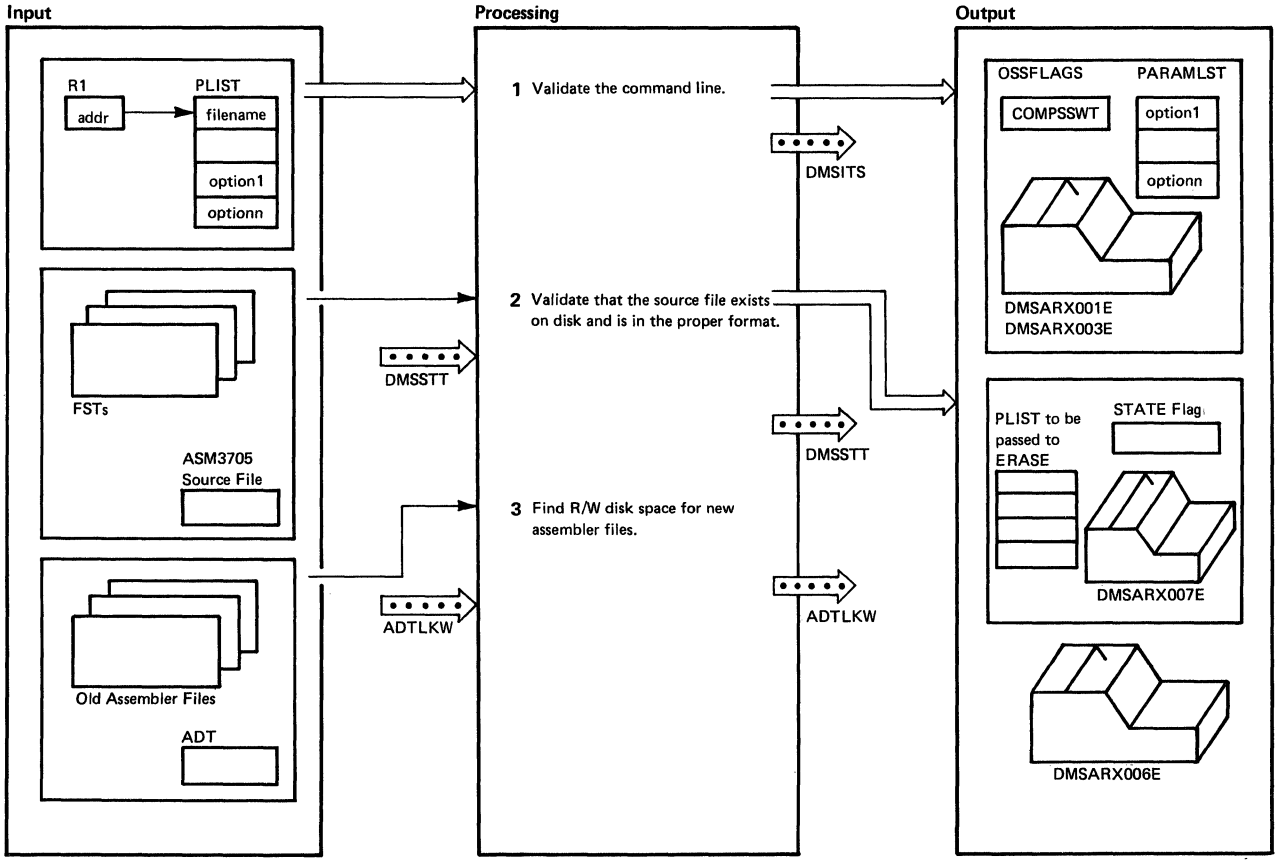
Notes		Module	Label	Ref
1	The filetype in the dummy FSCB is initialized to TEXT. Each linkage editor TEXT file has a filename consisting of the first 6 characters of the filename (or the entire filename if it is 6 characters or less) concatenated with L and a number.	DMSGRN	IEWL	
2	The //EXEC card is edited for the keyword PARM=. The linkage editor options are moved to the option field of the LKED command. EXEC continuation cards are ignored.	DMSGRN	IEWLJCLA	
3	The input file is scanned for the SYSLIN card. All cards scanned preceding the SYSLIN card must have * or // in the first positions. Otherwise, the error message DMSGRN078E Invalid card in input file *fn* is issued.	DMSGRN	IEWLJCL2	
4	The FSCBRD routine reads the input file and the FSCBWT routine writes it to the output file. The EDITIN routine scans for the keyword ENTRY. If the keyword ENTRY is found, the IEWLENT routine moves the entry name to the SAVENCP statement.	DMSGRN	IEWLSN10 WRTSIN IEWLENT	
5	The output file is closed by branching and linking to the FSCBCLOS routine. Close errors are ignored.	DMSGRN	IEWLSEOF FSCBCLOS	
6	The LKDFIRST bit in the PROCSW1 byte is tested. If it is off, it is set on and the filename of the input file is moved into the FILEDEF AND LKED commands. Also, the command count and address from STAKLKD1 are loaded into registers 1 and 2. If the LKDFIRST bit is on, the command count and address from STACKLKD2 are loaded into registers 1 and 2.	DMSGRN	LKDSTACK LKDSTAK1	

Figure 71. DMSGRN--Generating the Link Edit Files



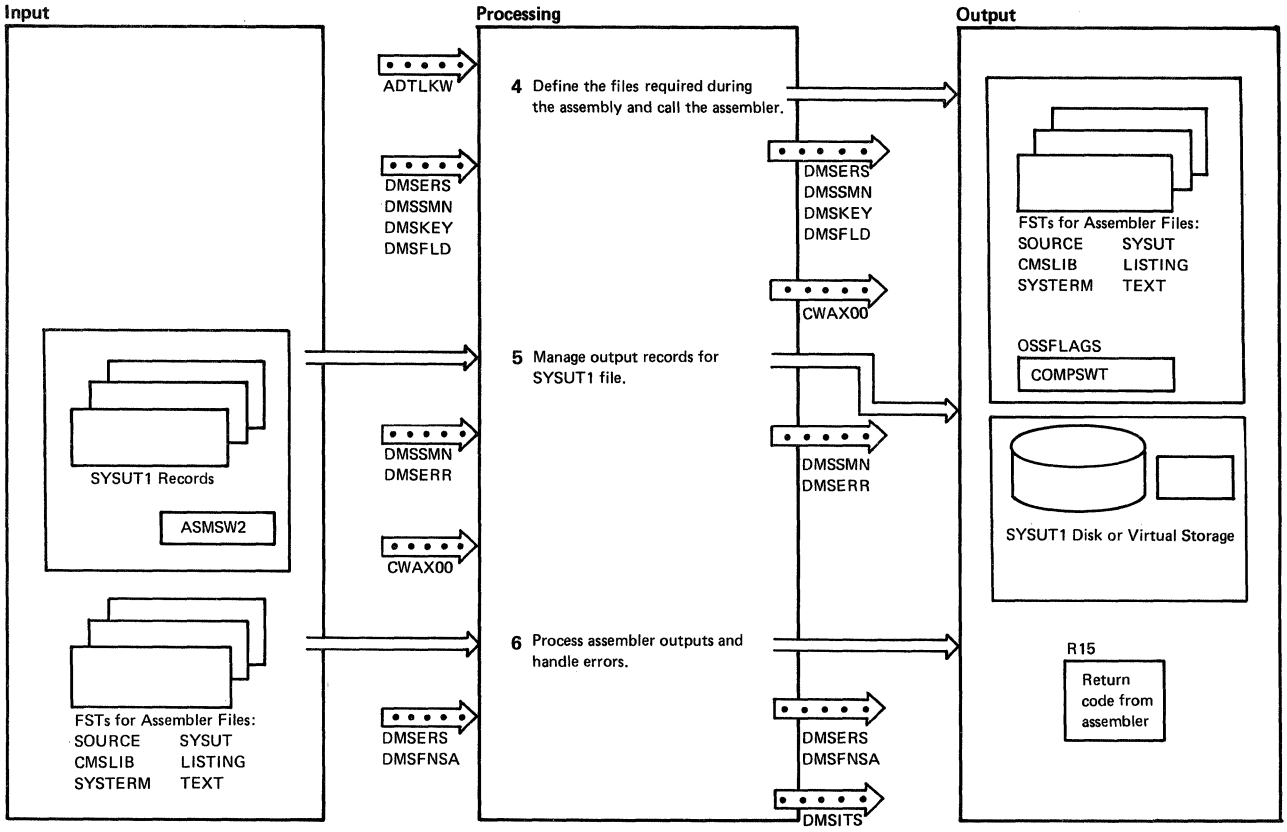
Notes		Module	Label	Ref
1	A filename must be specified. If it is not, the message DMSARN001E NO FILENAME SPECIFIED is issued and processing terminates.	DMSARN	DMSARN	
	The COMPSWT bit is set on in OSSFLAGS to indicate the 3705 assembler is running. The option list to be passed to the 3705 assembler is built.		SQUEEZE	
	If Batch is running, the message ASSEMBLING filename A1 is displayed and steps 2 and 3 are skipped.		SUIT15	
2	The STATE macro is issued to check that the input file exists and has fixed 80-character records. If the record format is wrong, the message DMSARN007E File *fn ft fm* is not fixed, 80-character records is issued and processing terminates.	DMSARN	SUIT25	
3	If the input file resides on a read/write disk, that disk is used to contain the text and listing files that are generated. If the input disk is an extension of a read/write disk, the parent disk is used. Otherwise, the A disk is used.	DMSARN	SUIT17	
4	All the old text, listing, and utility files for the current file are erased. Free storage is initialized and enough storage to contain the longest assemble path is obtained via a GETMAIN call.	DMSARN	CONTINUE	
	FILEDEFs are issued for SYSUT1, SYSUT2, SYSUT3, SYSIN, TEXT, SYSPUNCH (if the DECK option was specified), SYSPRINT (if the NOPRINT option was not specified), LISTING, and CMSLIB.		NOERASE	
5	Control is passed to IFKASM.	DMSARN	LIST2	
6	If the return code is not zero, one of the following messages is issued DMSARN004W WARNING MESSAGE ISSUED DMSARN008W ERROR MESSAGES ISSUED DMSARN016W TERMINAL ERROR MESSAGES ISSUED	DMSARN	RETURN	
	The output files are closed and the utility files SYSUT1, SYSUT2, and SYSUT3 are erased. All FCBs are cleared, OSSFLAGS is reset, and control returns to CMS.		SUIT19	

Figure 72. DMSARN--ASM3705 Command Processor (for the NCP/VS Release 2 and 3 Assembler).



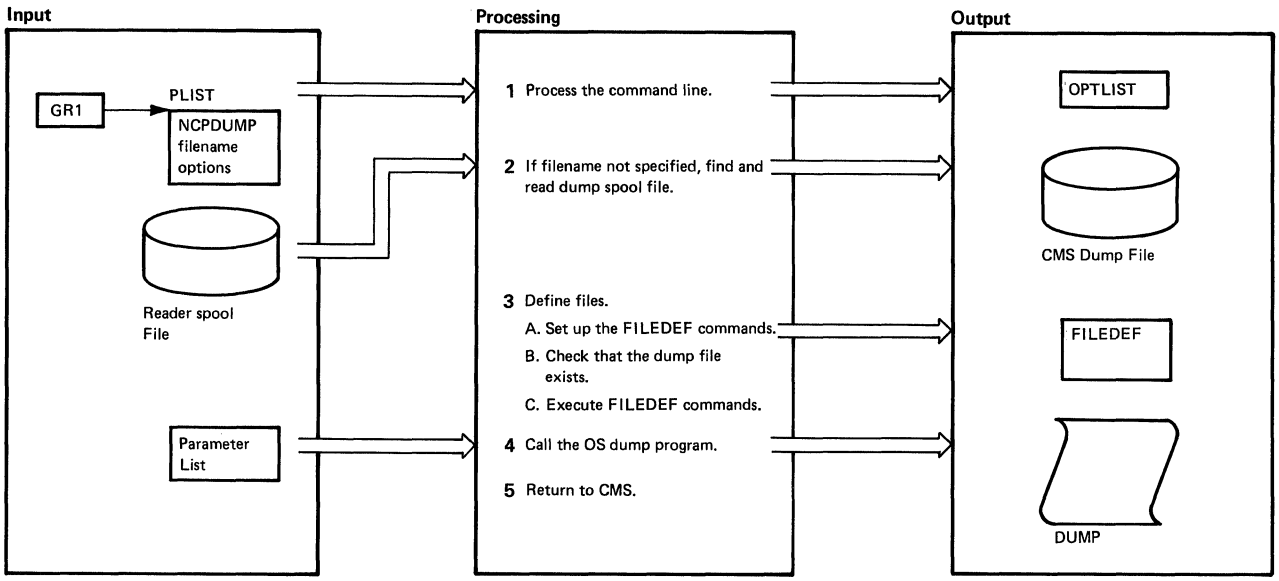
Notes		Module	Label	Ref
1	Validate the command line by ensuring that a filename has been specified and creating an assembler option list. If the filename is not specified, the message DMSARX001E NO FILENAME SPECIFIED is issued. The option list is built by scanning the command line, checking the options specified, and placing the valid entries in the PARMLIST table. If an invalid option is specified, the message DMSARX003E INVALID OPTION 'option' is issued and processing terminates. If the file is in proper format, processing continues at step 3.	DMSARX	OPTSCN	
2	Verify that the source file exists by issuing a STATE command (module DMSSTT). If the file exists but is not in proper format (80-character records), the message DMSARX007E FILE 'fn ASM3705' IS NOT FIXED, 80-CHAR. RECORDS is issued and processing terminates. If the file is in proper format, processing continues at step 3.	DMSARX	STATASM	
3	New files to be used during assembler processing (TEXT, LISTING, and SYSUT) can be obtained from three sources. If the input file resides on a R/W disk, that disk is used to contain the TEXT and LISTING files generated during the assembly. If the input file resides on an extension of the R/W disk, the parent disk is used. If neither of the above disks is a R/W disk, the user's A-disk is used. If no R/W disk can be obtained, the message DMSARX006E NO READ/WRITE DISK ACCESSED is issued and control returns to CMS via DMSITS.	DMSARX	FINDRW	

Figure 73 (Part 1 of 2). DMSARX--ASM3705 Command Processor (for the NCP/VS Release 4 Assembler).



Notes		Module	Label	Ref
4	DMSERS is called to erase the old TEXT, LISTING, and SYSUT files associated with the new input file. DMSSMN (GETMAIN) is called to obtain enough storage to contain the SYSUT1 work file.	DMSARX	ERASE	
	When disk space is obtained for the required assembler files and for the files CMS needs (SYSTERM and CMSLIB), FILEDEF commands are issued to convert all the files to CMS format. The assembler is then called and begins processing.		FILEDEF LOADASM	
5	If possible, all SYSUT1 records are kept in virtual storage during an assembly. However, when virtual storage is exhausted, records are written to disk.	DMSARX	ASMPROC	
	If the records must be written to disk, they are formatted to fit DASD requirements and moved to disk a record at a time.		SYSWTX	
6	All SYSUT files used during the assembly are erased via a call to DMSERS, DMSFNSA is called to close all files and DMSFLD is called to clear all FILEDEFs not defined with the PERM option. COMPSWT in OSSFLAGS is turned off to indicate that the assembler is no longer processing, the auxiliary directory list is released, and control returns to CMS via DMSITS.	DMSARX	ERASUTS RETURN	

Figure 73 (Part 2 of 2). DMSARX--ASM3705 Command Processor (for the NCP/VS Release 4 Assembler).



Notes			Module	Label	Ref
1	If the second parameter in the input line starts with DUMP, the name of the CMS file is saved in the output FSCB. The appropriate options are marked in the OPTLIST. If there are no options specified, FORMAT, no MNEMONIC, and no ERASE are assumed. If an invalid option is specified, the following message is generated DMKRND863E INVALID PARAMETER 'xxxxxxxx' and control returns to CMS with a return code of 24.		DMKRND	NCPDUMP TESTOPT	
2	If the name of a CMS dump file was not specified, DMKRND assumes the dump file is in the reader. The filename of the output file is set to DUMP00 through DUMP09 and the STATE macro is issued until a dump file is found. If an available name is not found, the following message is generated. DMKRND851I TEN DUMP FILES ALREADY EXIST and control returns to CMS with a return code of 22.		DMKRND	LOOKLOOP	
	The reader is spooled class E and the spool file is read via a DIAGNOSE instruction. The records are deblocked and written to the CMS dump file. The read/write loop continues until the real spool file DIAGNOSE instruction returns a nonzero return code. When the end of the file is reached, the message 'DUMPnn NCPDUMP' FILE CREATED is issued, the spool file is closed, and processing continues. If the reader was empty or if a read error occurs, an error message is issued. DMKRND853I NO DUMP FILES EXIST DMKRND850I UNABLE TO READ DUMP FROM READER			READNXT DUMPWRT	
3					
	A	The name of the CMS dump file is put in the SYSUTZ and SYSIN FILEDEFs and in the control statement skeleton for the IPLDUMP processor.	DMKRND	STRDUMP	
	B	The STATE macro is issued to check that the CMS dump file exists. If an error is returned, the following message is generated DMKRND861E FILE 'DUMPnn NCPDUMP' NOT FOUND The SYSIN record is created, using the specified user options, any old SYSIN file is erased, and the new SYSIN file is written to the DUMPnn SYSIN file. If the record cannot be written, the message DMKRND870I UNABLE TO CREATE CONTROL FILE FOR IPLDUMP is issued and control returns to CMS.			
	C	The following commands are issued to simulate an OS interface. FILEDEF SYSUT2 DISK DUMPnn NCPDUMP A1 (XTENT 513 NOCHANGE FILEDEF SYSIN DISK DUMPnn SYSIN A1 FILEDEF SYSPRINT PRINTER		LINKDMP	
4	DMIRND loads register 1 with the address of a dummy parameter list and links to IFLDUMP. If the return code from IFLDUMP is not zero, it is passed to CMS.		DMKRND		
5	If the return code from IFLDUMP is zero and ERASE has been requested, the DUMPnn file is erased, and the following message is generated 'DUMPnn NCPDUMP' FILE ERASED		DMKRND		

Figure 74. DMKRND--NCPDUMP Command Processor

Program Organization

This section describes the following 3704/3705 command processing modules:

- DMKRND--NCPDUMP command processor
- DMSARN--ASM3705 command processor (for NCP/VS Release 2 and 3 Assembler)
- DMSARX--ASM3705 command processor (for NCP/VS Release 4 Assembler)
- DGRN--GEN3705 command processor
- DMSNCP--SAVENCP command processor

DMKRND

The interface to the OS/360 3705 dump program.

Entry Point

DMKRND

Attributes

Runs in a CMS virtual machine

Entry Conditions

Reg	Use
R1	Address of parameter list
R13	Address of savearea
R14	Return address
R15	CSECT base register

Register Usage

Reg	Use
R0-10	Work registers
R11	Address of FSCBDSECT
R12	CSECT base register
R13	Address of savearea
R14	Linkage register
R15	Return code

Routines Called

Routine	Purpose
IFLDUMP	To format and print the dump

External References

None

Data Areas

FSCB

Exit Conditions

Reg	Use
R12	CSECT base address
R13	Address of input savearea
R14	Return address
R15	Return code

DMSARN

The interface between CMS and the 3704/3705 Assembler (IFKASM).

Entry Points

DMSARN	To process the ASM3705 command
ASMHAND	To handle any I/O activity pertaining to the SYSUT2 file during the assembly

Attributes

Disk resident

Entry Conditions

At DMSARN

Reg	Use
R1	Address of the parameter list
R14	Return address
R15	Address of the entry point

At ASMHAND

Reg	Use
R1	Address of the DECB
R2	Address of the DCB
R8	Address of the OPSECT
R11	Address of the FCBSECT
R14	Return address
R15	Address of the entry point

Register Usage

Reg	Use
R0-1	Work registers
R3	Base register
R4-5	Work registers
R6	Return address to caller
R7-9	Work registers
R10	Constant 8
R12-13	Work registers
R14	Linkage register
R15	Error code

Routines Called

Routine	Purpose
DMSERSA	To erase old files
DMSSMNE	To initialize storage pointers
DMSSTTA	To locate the file
IFKASM	To assemble the 3704/3705 control program

External References

Routine	Purpose
FREEMAIN	To return free storage
GETMAIN	To obtain free storage
NUCON	The nucleus constant area
TYPE	To send messages to the terminal

Data Areas

None

Exit Conditions

Contents of register 15 indicate results of processing.

Return Code	Meaning
0	No errors
4	Minor errors detected during assembly, successful program execution is probable
8	Errors detected during assembly, unsuccessful program execution is possible
12	Serious errors detected during assembly, unsuccessful execution is probable
16	Critical errors detected during assembly, unsuccessful execution is probable
20	Catastrophic errors detected during assembly, partial or complete assembly canceled
24	Invalid option, no filename
28	File not found
32	Invalid record length for ASM3705 file
36	No read/write disks accessed

DMSARX

The interface between the ASM3705 command and the 3704/3705 Assembler (CWAX00).

Entry Points

DMSARX

ASMPROC	SYSUT1 processing routine
TERMPROC	Terminal output processing routine

Attributes

Executes in user area

Entry Conditions

Reg	Use
R1	Address of the parameter list
R14	Return address
R15	Address of the entry point (DMSARX)

Register Usage

Reg	Use
R0	NUCON addressability
R1	Address of all PLISTs
R2	Work register
R3	Work register
R4	GETMAIN/FREEMAIN amount
R5	Work register
R6	GETMAIN/FREEMAIN address
R7	ASMPROC address
R8	Work register
R9	Work register
R10	Linkage register
R11	FCB address during ASMPROC
R12	Base register
R13	Save area address
R14	Return register from calls
R15	Assembler root address and return error code

Routines Called

Routine	Purpose
DMSCRD	Read SYSPARM from console
DMSCWR	Display SYSPARM message to console
DMSFLD	FILEDEF all assembler files
DMSFNS	Close all assembler files
DMSKEY	Control nucleus protect key
DMSERR	Display all error messages
DMSERS	Erase old assembler files
DMSSLN	Load the assembler phases
DMSSMN	Control storage pointers (GETMAIN/FREEMAIN)
DMSSTT	Verify disk file existence
DMSLADAD	SET/RESET the FST chain for auxiliary directory

Routine	Purpose
CWAX00	3705 assembler (XF) root segment

External References

ADT
CMSCB
DMSARD
FSTB
IO
NUCON

Data Areas

Area	Purpose
DDNAME	Names of CMS ddnames for assembler
OPTLIST	Option list passed to the assembler
OPDEF	(Macro label) names and abbreviations of all options
PARAMLST	Parameter list for assembler
UTENTRY	In-core SYSUT1 record area
UTHEAD	Header area for in-core records
OPTAB$	List of pointers to option table entries
SAVEAREA	SAVEAREA

Exit Conditions

NORMAL

GPR15=0 No error

ERROR

Code	Condition
GPR15=24	Invalid option, no filename specified
GPR15=28	File not found
GPR15=32	File not fixed, 80 char. records
GPR15=36	No read/write disks accessed
GPR15=40	Fileid conflict, device invalid for input

Return Code	Meaning
0	No errors
4	Minor errors detected during assembly, successful program execution is probable

Return Code	Meaning
8	Errors detected during assembly, unsuccessful program execution is possible
12	Serious errors detected during assembly, unsuccessful execution is probable
16	Critical errors detected during assembly, unsuccessful execution is probable
20	Catastrophic errors detected during assembly, partial or complete assembly canceled
24	Invalid option, no filename
28	File not found
32	Invalid record length for ASM3705 file
36	No read/write disks accessed

DMSGRN

Edits the Stage 2 input for the 3704/3705 control program generation, builds the 3704/3705 assembler files and linkage editor text files, and builds an EXEC macro file.

Entry Point

DMSGRN

Attributes

Runs in a CMS virtual machine

Entry Conditions

Reg	Use
R1	Address of the input parameter list
R13	Address of the savearea
R14	Return address
R15	CSECT base address

Register Usage

Reg	Use
R0-10	Work registers
R11	Base register 2
R12	Base register 1
R13	Address of the savearea
R14	Linkage register

Reg	Use
R15	Return code

Routines Called

None

External References

None

Data Areas

FSCB

Exit Conditions

Reg	Use
R12	Base address
R13	Address of input savearea
R14	Return address
R15	Return code

DMSNCP

Reads a 3705 control program module (EP or NCP) in OS load module format and writes a page-format core-image copy on the VM/SP system volume.

Entry Point

SAVENCP

Attributes

Serially reusable, executes in a CMS virtual machine

Entry Conditions

Reg	Use
R1	Address of the input parameter list

Register Usage

Reg	Use
R0	Work register
R1	Address of parameter list and word register
R2	Pointer to input record and work register
R3	Length of input record and work register
R4-6	Work registers
R10	Address of the input file DCB during the read, then the address of the control program core image.
R11	Address of the CCPARM parameter list
R12	Base register
R13	Address of the savearea
R14	Linkage register
R14	Linkage and work register

Routines Called

DMKSNC via Diagnose Code X'50' to write the core image of the 3704/3705 control program and parameters on disk

External References

None

Data Areas

CCPARM

Exit Conditions

Reg	Use
R15	Return code

Directory

Following is a list of CP and CMS modules that process the commands that generate the 3704/3705 control program and process the 3704/3705 storage dumps.

Module	Description
DMKRND	NCPDUMP command processor
DMSARN	ASM3705 command processor
DMSARX	ASM3705 command processor
DMSGRN	GEN3705 command processor
DMSNCP	SAVENCP command processor

Following are the label directories, lists of the major labels in each of the command processors. In addition to the label, the module (if more than one is involved), associated method of operation diagram, and a brief description are included in the list.

The NCPDUMP Command Processor (DMKRND)

Label	Figure	Description
DUMPWRT	74	Writes the output file
LINKDMP	74	Links to the OS dump service program, IFLDUMP
LOOKLOOP	74	Checks the reader for a valid CMS dump file
NCPDUMP	74	Starts processing the NCPDUMP command
READNXT	74	Reads the dump spool file
STRTDUMP	74	Builds the control file for the IFLDUMP processing routine
TESTOPT	74	Processes the options on the NCPDUMP command line

The ASM3705 Command Processor (DMSARN)

Label	Figure	Description
CONTINUE	72	Erases old files and gets enough storage for the assembler to execute in
DMSARN	72	Entry point for the ASM3705 command processor
LIST2	72	Calls the 3705 Assembler (IFKASM)
NOERASE	72	Issues FILEDEFs for the necessary assembler files
RETURN	72	Returns control to CMS
SQUEEZE	72	Checks that the input file exists
SUIT15	72	If running in a batch machine, sends ASSEMBLING filename A1 message
SUIT17	72	Finds a read/write disk for writing text and listing files
SUIT19	72	Closes the output files and erases the utility files

Label	Figure	Description
SUIT25	72	Checks the format of the input file

The ASM3705 Command Processor (DMSARX)

Label	Figure	Description
ERASE	73	Erases old files
DMSARX	73	Entry point for the ASM3705 command processor
FILEDEF	73	Issues FILEDEFs for the necessary assembler files
FINDRW	73	Finds a read/write disk for writing text and listing files
LOADASM	73	Load the 370X Assembler root
OPTSCN	73	Validates command line
RETURN	73	Returns control to CMS
VERIFY	73	Checks that the input file exists

The GEN3705 Command Processor (DMSGRN)

Label	Figure	Description
ASMSTAK	70	Stacks the required 3705 Assembler commands in the Stage 2 EXEC macro file
ASMSTAK2	70	Puts the name of the output assembler file in the ASM3705 and EDIT commands
ASMSTAK4	70	Puts the SYSPUNCH membername in the TXTLIB command
ASMSTAK6	70	Puts the number of commands and the address of the first command into registers 1 and 2
CLOSTACK		Builds the SAVENCP command
EDITIN		Edits the input records for keywords
FINDASM	69	Checks for assembler input
FINDIEWL	69	Checks for linkage editor input
FSCBCLOS	70	Closes the output file
FSCBRD		Reads the input file
FSCBWT		Writes the output file
GENMSG		Generates error messages
IEWL	71	Main processing routine for generating linkage editor commands
IEWLENT	71	Scans for the keyword ENTRY
IEWLJCLA	71	Edits the //EXEC statement
IEWLJCL2	71	Scans for the //SYSLIN statement
IEWLSEOF	71	Branches and links to FSCBCLOS to close the linkage editor output file
IEWLSIN		Processes SYSLIN information
IEWLSN10	71	Branches and links to FSCBRD to read the linkage editor input file
IFKASM	70	Main processing routine for generating 3705 assembler files
IFKASMA0	70	Branches and links to the FSCBRD and FSCBWT routines to read the input file and write the output file
IFKASMK0	70	Branches and links to the FSCBCLOS routine to close the output

Label	Figure	Description
		assembler files
IFKASM10	70	Scans for the SYSPUNCH statement
IFKASM34	70	Scans for the DSN= or DSNAME= keyword on the SYSPUNCH statement
IFKASM40	70	Scans for the SYSIN statement
LKDSTACK	70	Builds the LKED commands and the FILEDEF for their file
LKDSTAK1	71	Loads registers 1 and 2 with the number of commands and the address of the first linkage editor command
OPTEND	69	Checks that the input file exists
OPTIONS1	69	Scans the input options
PRIMEDIT	69	Scans for a valid //EXEC statement
PROCEND1	69	Closes the EXEC file
PROCEND2	69	Adds the SAVENCP command to the EXEC macro file
PROCWT		Writes commands to the Stage 2 EXEC processor file
RETURN1	69	Returns control to CMS
STACK30	69	Writes the linkage editor and assembler statements to the EXEC macro file
START	69	Starts the GEN3705 command processing
WRTSIN	70	Branches and links to the FSCBWT routine to write the linkage editor output file

The SAVENCP Command Processor (DMSNCP)

Label	Figure	Description
CESDCHVT	67	Finds the channel vector table
CESDENT	67	Saves the entry point
CHEKVMV	68	Checks that the specified options are compatible
CLOSE	67	Closes the input file
CONTROL	67	Moves the text records from the input buffer to the core image buffer
ENDPARMS	67	Opens the input file and searches for the member
ERR21	67	Checks for the entry point record
ERR66	67	Checks for premature end of file or invalid control records
EXIT	67	Returns control to CMS
SAVECCP	67	Issues the Diagnose X'50' instruction to have DMKSNC do the actual saving
SAVENCP	67	Entry point for the SAVENCP command processor
SCANCEP	68	Updates the CCPARM parameter list for EP and PEP control programs
SCANDEV		Scans for devices
SCANLINE		Scans for teleprocessing lines
SCANNCP	68	Updates the CCPARM parameter list for NCP and PEP control programs

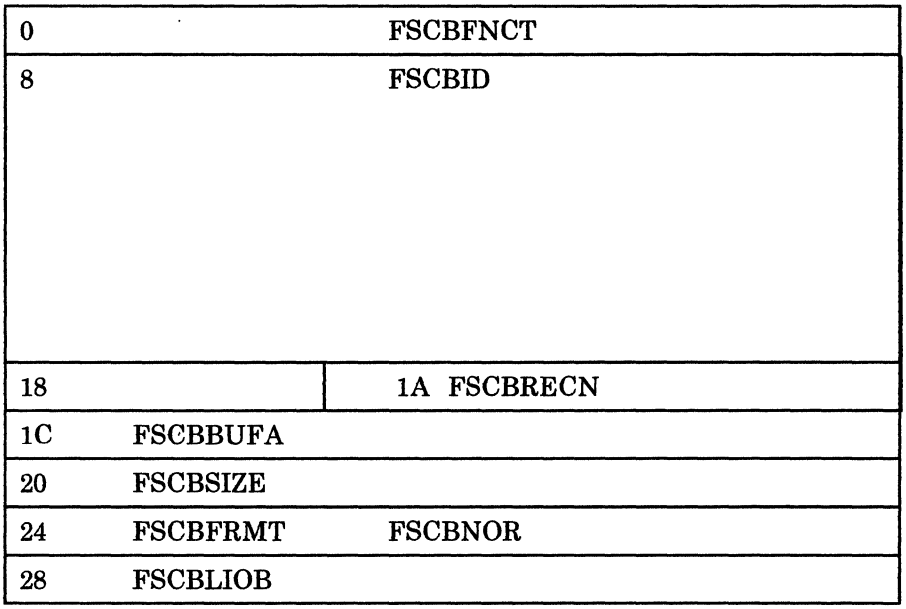
Data Areas

The following data areas are used by the 3704/3705 command processor modules:

- Active Disk Table (ADT)
- Communications Controllers Parameter List (CCPARM)
- File System Control Block (FSCB)
- Input/Output Block (IOBLOK)
- Network Interface Control Block (NICBLOK)
- Real Device Block (RDEVBLOK)
- Spool File Block (SFBLOK)
- Virtual Machine Block (VMBLOK).

All the above data areas except the FSCB are described in the *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* and *VM/SP Data Areas and Control Block Logic Volume 2 (CMS)*. The FSCB is described in Figure 75.

File System Control Block



Displacement Hex	Dec	Field Name			Description
0	0	FSCBFNCT	DS	CL8	Control field for I/O function
8	8	FSCBID	DS	0CL18	File Identifier
8	8	FSCBFN	DS	CL8	Filename
0	16	FSCBFT	DS	CL8	Filetype
18	24	FSCBFM	DS	CL2	Filemode
1A	26	FSCBRECN	DS	H	Relative record number
1C	28	FSCBBUFA	DS	A	Buffer address
20	32	FSCBSIZE	DS	F	Buffer size
24	36	FSCBFRMT	DS	CL2	File format
26	38	FSCBNOR	DS	0H	Number of records to be read
28	40	FSCBLIOB	DS	A	

Figure 75. File System Control Block

Diagnostic Aids

Listed below are the messages and abnormal termination codes.

The NCPDUMP Command Processor (DMKRND)

Message Code	Label	Figure	Message Text
DMKRND850I	DUMPWRT	74	UNABLE TO READ DUMP FROM READER (Return Code = 21)
DMKRND851I	LOOKLOOP	74	TEN DUMP FILES ALREADY EXIST (Return Code = 22)
DMKRND852I			FATAL I/O ERROR WRITING DUMP
DMKRND853I	DUMPWRT	74	NO DUMP FILES EXIST (Return Code = 23)
DMKRND861E	STRTDUMP	74	DUMP FILE filename NOT FOUND (Return Code = 28)
DMKRND863E	TESTOPT	74	INVALID PARAMETER - parameter (Return Code = 24)
DMKRND870I	STRTDUMP	74	UNABLE TO CREATE CONTROL FILE FOR IPLDUMP (Return Code = 16)
	DUMPWRT	74	'DUMPnn NCPDUMP' FILE CREATED
	LINKDMP	74	'DUMPnn NCPDUMP' FILE ERASED

The ASM3705 Command Processor (DMSARN)

Message Code	Label	Figure	Message Text
DMSARN001E	DMSARN	72	NO FILENAME SPECIFIED
DMSARN002E			[INPUT] {FILE[(s)]/DATA SET} ['fn[fm]]' NOT FOUND [OVERLAY]
DMSARN003E			INVALID OPTION 'option'
DMSARN004W	RETURN	72	WARNING MESSAGES ISSUED
DMSARN006E			NO READ/WRITE ['A'] DISK ACCESSED [FOR 'fn ft']
DMSARN007E	SUIT25	72	FILE 'fn ft fm' [IS] NOT FIXED, 80-CHAR. RECORDS
DMSARN008W	RETURN	72	ERROR MESSAGES ISSUED
DMSARN012W	RETURN	72	SEVERE ERROR MESSAGES ISSUED
DMSARN016W	RETURN	72	TERMINAL ERROR MESSAGES ISSUED
DMSARN109S			VIRTUAL STORAGE CAPACITY EXCEEDED

The ASM3705 Command Processor (DMSARX)

Message Code	Label	Figure	Message Text
DMSARX001E	OPTSCN	73	NO FILENAME SPECIFIED
DMSARX002E	NEWFILE	73	[INPUT] {FILE [(s)]/DATA SET} ['fn[fm]]' NOT FOUND [OVERLAY]
DMSARX003E	OPTSCN	73	INVALID OPTION 'option'
DMSARX006E	FINDRW	72	NO READ/WRITE ['A'] DISK ACCESSED [FOR 'fn ft']
DMSARX007E	STATASM	73	FILE 'fn ft fm' [IS] NOT FIXED, 80-CHAR. RECORDS
DMSARX038E	DOFDEF	73	FILEID CONFLICT FOR DDNAME 'ASM3705'
DMSARX052E	MOVEKEY	73	MORE THAN 100 CHARS OF OPTIONS SPECIFIED
DMSARX070E	DMSARX	73	INVALID {PARAMETER 'parameter'/ARGUMENT 'argument'}
DMSARX074E	DMSARX	73	ERROR [RE]SETTING AUXILIARY DIRECTORY
DMSARX075E	NOTDSK	73	DEVICE 'device' INVALID FOR {INPUT/OUTPUT}

The GEN3705 Command Processor (DMSGRN)

Message Code	Label	Figure	Message Text
DMSGRN002E	OPTEND	69	File *fn [ft [fm]]* not found
DMSGRN003E	OPTIONS1	69	Invalid option: *option*
DMSGRN007E			File *fn ft fm* is not fixed, 80-character records
DMSGRN048E	OPTEND	69	Invalid mode *mode*
DMSGRN054E			Incomplete fileid specified
DMSGRN078E	PRIMEDIT	69	Invalid card in input file *fn*
	IFKMAS40	70	'xxx... x'
	IEWLJCL2	71	

The SAVENCP Command Processor (DMSNCP)

Message Code	Label	Figure	Message Text
DMSNCP001E	SAVENCP	67	NO FILENAME SPECIFIED (Return Code = 24)
DMSNCP002E	ENDPARMS	67	[INPUT/OVERLAY] {FILE[(s)]/DATA SET} ['fn[fm]]' NOT FOUND (Return Code = 28)
DMSNCP003E	SAVENCP TESTOP		INVALID OPTION 'option' (Return Code = 24)
DMSNCP013E	DMS0001A	67	MEMBER 'name' NOT FOUND IN LIBRARY ['fn ft fm'/'libname'} (Return Code = 4)

Message Code	Label	Figure	Message Text
DMSNCP021E	CONTROL	67	ENTRY POINT 'name' NOT FOUND (Return Code = 40)
DMSNCP025E	SCANCEP SCANNCP	68	INVALID DATA IN 370X CONTROL PROGRAM (Return Code = 16)
DMSNCP045E	CLOSE		UNSUPPORTED 370X CONTROL PROGRAM TYPE (Return Code = 16)
DMSNCP056E	NOTLAST	67	FILE 'fn ft fm' CONTAINS INVALID {NAME/ALIAS/ENTRY/ESD} RECORD FORMATS
	CLOSE ERR66		(Return Code = 32)
DMSNCP099W	CHEKVMV	68	GENERATION PARAMETERS INCOMPATIBLE WITH VM/370 (Return Code = 99)
DMSNCP109S	CONTROL NOTLAST	67	VIRTUAL STORAGE CAPACITY EXCEEDED (Return Code = 104)

Index

Chapter 8. ZAP Service Program

ZAP Service Program

Introduction

The ZAP service program (DMSZAP) executes under the control of CMS via the ZAP and ZAPTEXT commands. The ZAP command performs functions for LOADLIB, TXTLIB, and MODULE files residing on direct access storage devices. The ZAPTEXT command performs functions for individual text files and internally invokes ZAP. For a complete description of the ZAP and ZAPTEXT commands, see the *VM/SP Operator's Guide*

The functions that ZAP and ZAPTEXT can do are:

- Dump
- Verify
- Replace.

In addition, ZAPTEXT can also perform the EXPAND function.

Dump

The dump function reads all or part of a specified CSECT, or an entire member or module, formats the dump, and prints it at the system printer (133-character lines, each containing 32 bytes in hexadecimal, plus the translation) or displays it at the terminal (80-character lines, each containing 16 bytes in hexadecimal, plus the translation). If more than one CSECT is dumped, the CSECT name appears before each dump.

Verify

The verify function compares specified data with the data at a specified address in a CSECT. If the data is the same, a replace operation (if one is specified) is permitted; otherwise, an error message is issued.

Replace

The replace function replaces data at a specified address in a CSECT with the data specified in a control record. The changed record is then written back to the file.

Expand

The ZAPTEXT service program uses the EXPAND command to add space to a program in object deck form. However, the ZAP service program ignores the EXPAND command.

Method of Operation

The method of operation diagrams describe the execution of the ZAP program and show the processing associated with:

- Verifying and replacing data in a CSECT
- Dumping a CSECT, member, or module.

The relationship of the method of operation figures is shown in Figure 76.

Figure 77 on page 200 describes the execution of the ZAP program.

Figure 78 on page 201 shows the ZAP command and control record processing.

Figure 79 on page 202 describes the processing of the DUMP function.

Figure 80 on page 203 and Figure 81 on page 204 describe the processing for modifying data in a CSECT.

Figure 82 on page 205 and Figure 83 on page 206 describe how the proper CSECT is located for dumping or modifying.

Figure 84 on page 207 shows how a file is read for dumping or modifying.

Figure 85 on page 208 describes how a dump is printed.

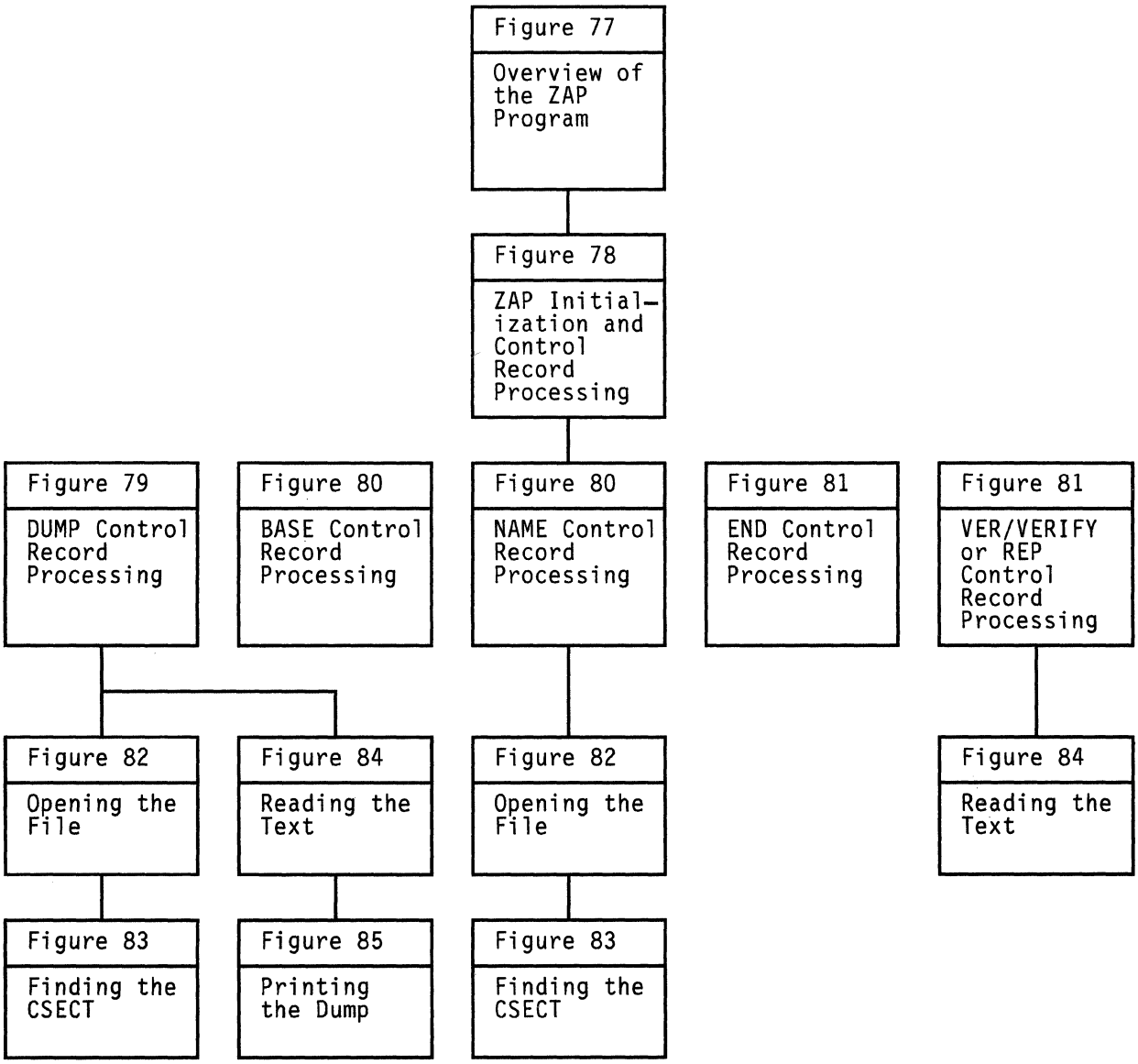
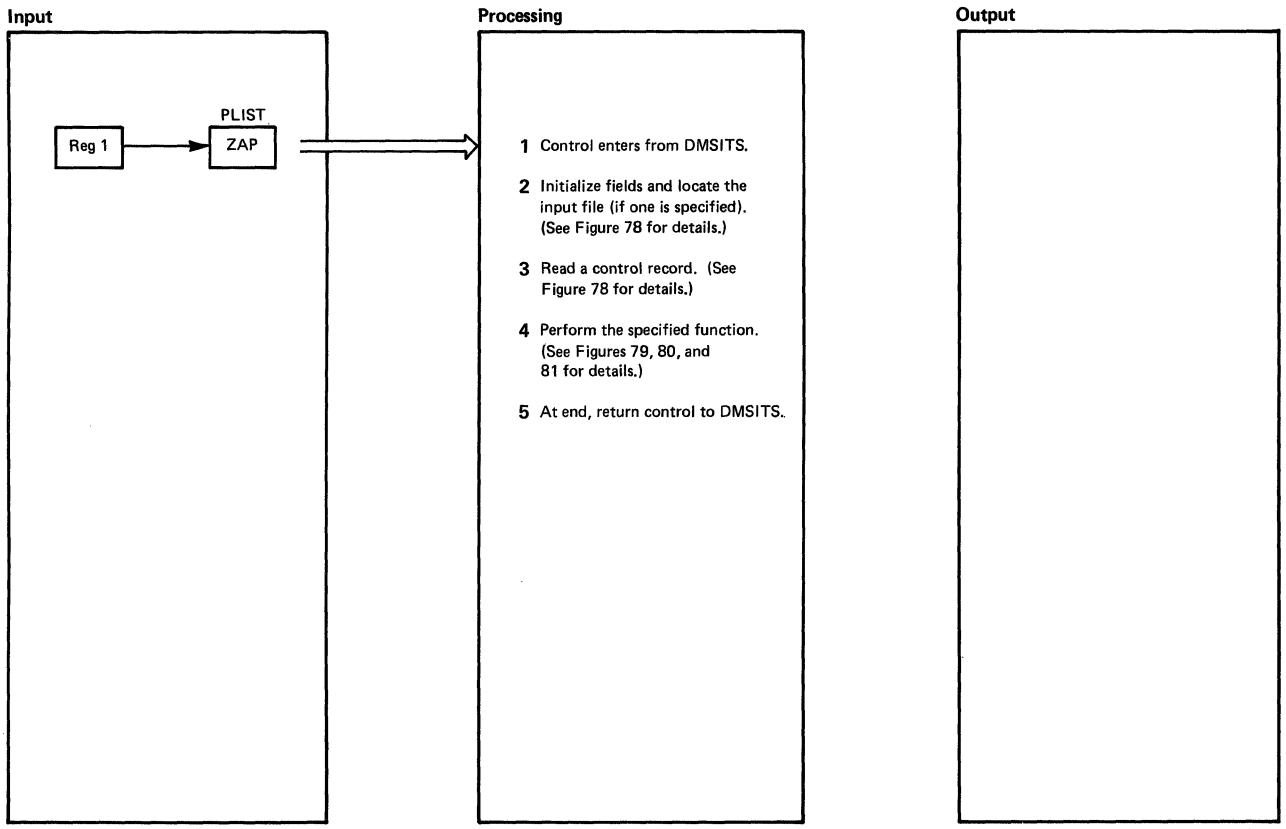


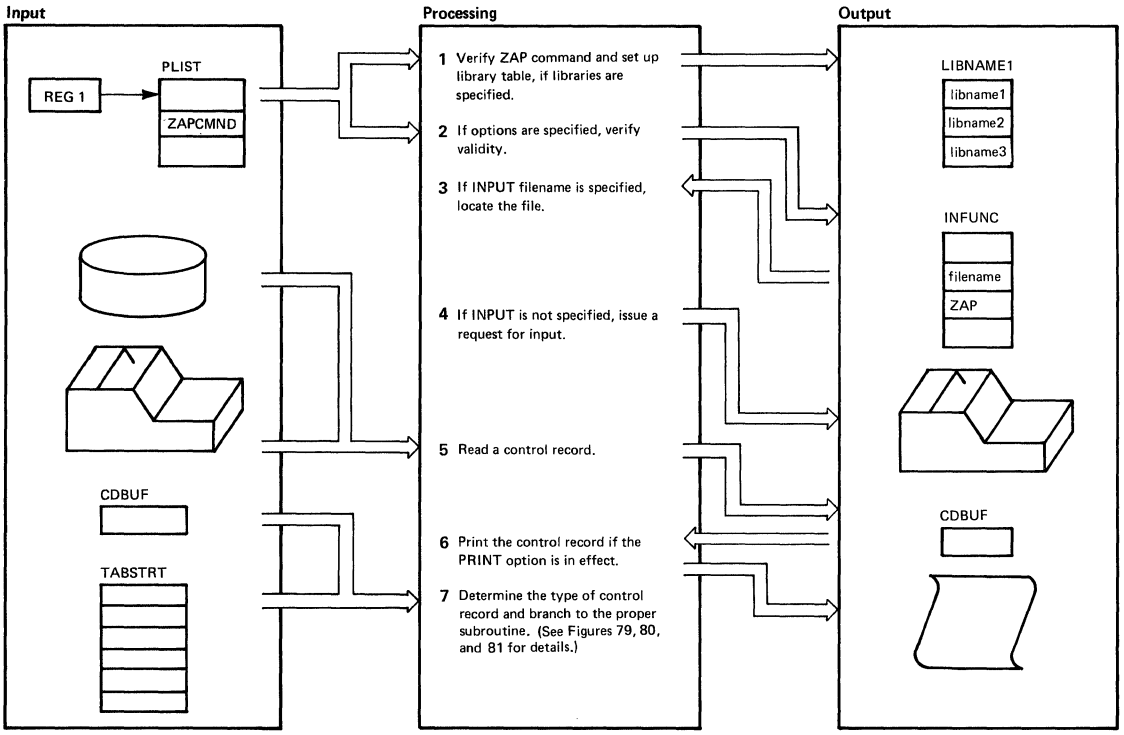
Figure 76. Key to the ZAP Program Method of Operation Figures

ZAP Service Program



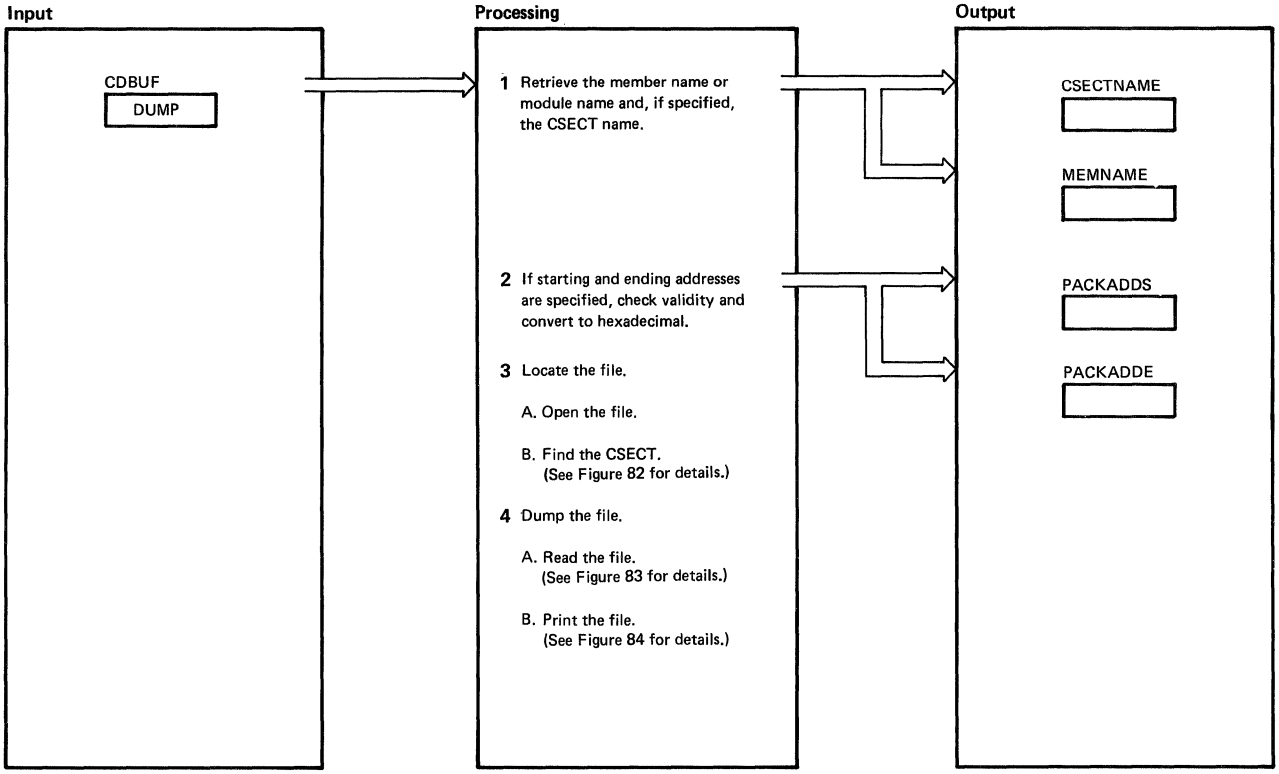
Notes		Module	Label	Ref
1	Control enters DMSZAP from DMSITS. Register 1 points to a PLIST that contains the type of file to be operated on, libraries to be used if applicable, and controls for input and output operation.	DMZAP	DMSZAP	
2	Initialize fields and pointers and verify input and output options. Locate the input file if an input file is specified. Otherwise, request input from the terminal.	DMSZAP	SCANLINE INITOPEN FDEFINP	
3	Read a control record. Find the routine needed to perform the function specified by searching a table of control record keywords.	DMSZAP	READINP	
4	Perform the specified function. At its end, return control to READINP to read another control record.	DMSZAP		
5	When the END control record is read, return control to DMSITS.	DMSZAP		

Figure 77. Overview of the ZAP Program



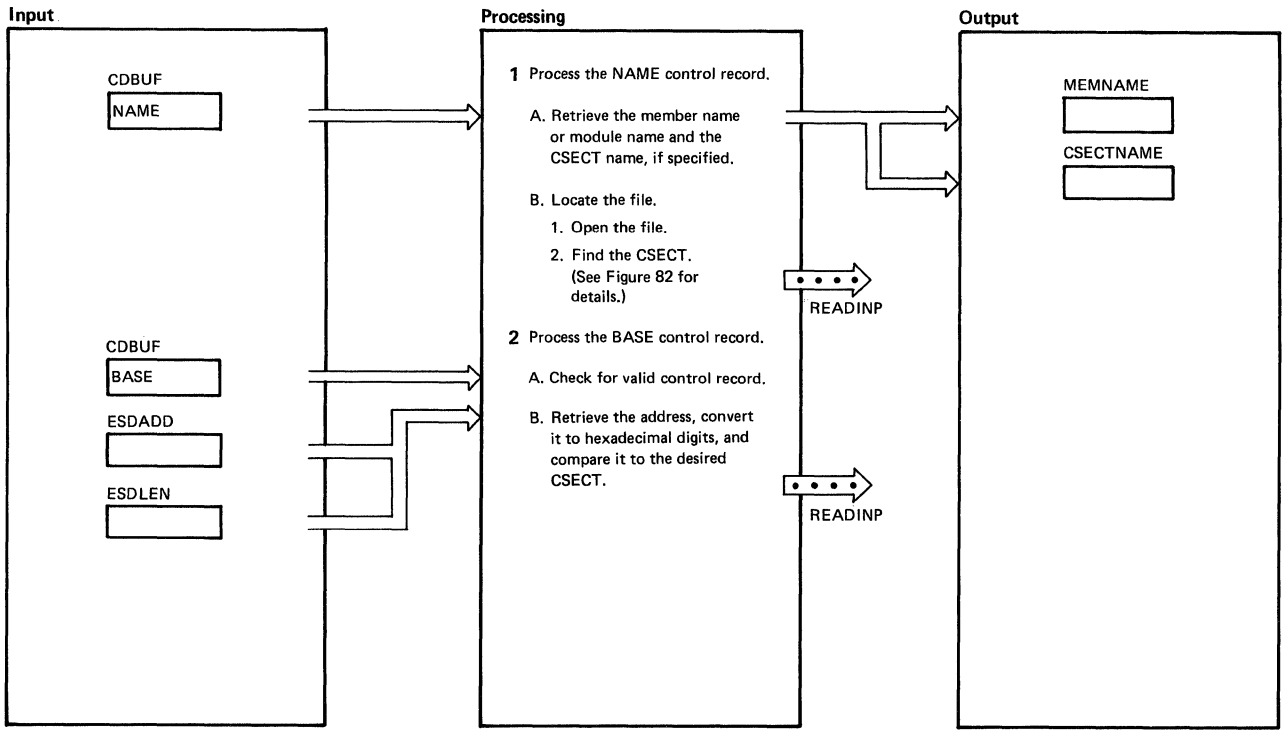
Notes		Module	Label	Ref
1	Verify the operands in the ZAP command. If TXTLIB or LOADLIB is specified, move the library name (up to three) into LIBNAME1. If no library name was specified, issue the message: DMSZAP001E No filename specified Other messages that may be issued if the command line is in error are: DMSZAP014E Invalid function *function* DMSZAP047E No function specified DMSZAP070E Invalid parameter *parameter*	DMSZAP	SCANLINE STLIB	
2	If options are specified, check for validity. If mutually exclusive options or invalid options are specified, issue the message: DMSZAP003E Invalid option: *option*	DMSZAP	CHKOPT	
3	If INPUT filename is specified, move filename into INFUNC. Issue STATE to locate the file. If this file cannot be found, the message: DMSZAP002E File *[fn [ft [fm]]]* not found	DMSZAP	INPTOPT FDEFINP	
4	If INPUT is not specified, display ENTER: to request ZAP control records to be entered from the terminal.	DMSZAP	READINP RDCARD	
5	Read the control record either from the terminal (RDCARD routine) or from the specified INPUT file (RDCARD2 routine). Save the control record in CDBUF.	DMSZAP	RDCARD RDCARD2	
6	Print the control record on the SYSOUT printer if the PRINT option is in effect.	DMSZAP	WRCARD	
7	Check the control record for a valid keyword. If the statement is blank, or the first word is EXPAND, or the first character is an asterisk, return control to READINP (step 4).	DMSZAP	SCANKEY1	
	Otherwise, compare the keyword to keyword tables whose formats are: bytes 1-8 keyword bytes 9-12 keyword routine		TABLOOK	
	Valid keywords and the diagrams in which their routines are described are: **Keyword** **Figure** DUMP Figure 79 NAME Figure 80 BASE Figure 80 VER Figure 81 VERIFY Figure 81 REP Figure 81 END Figure 81		NAMFOUND	
	If a match is found, go to the appropriate routine. If no match is found, issue the message: DMSZAP201W INVALID CONTROL RECORD OR NO GO SWITCH SET and return control to READINP (step 4).		INVEREP	

Figure 78. ZAP Initialization and Control Record Processing



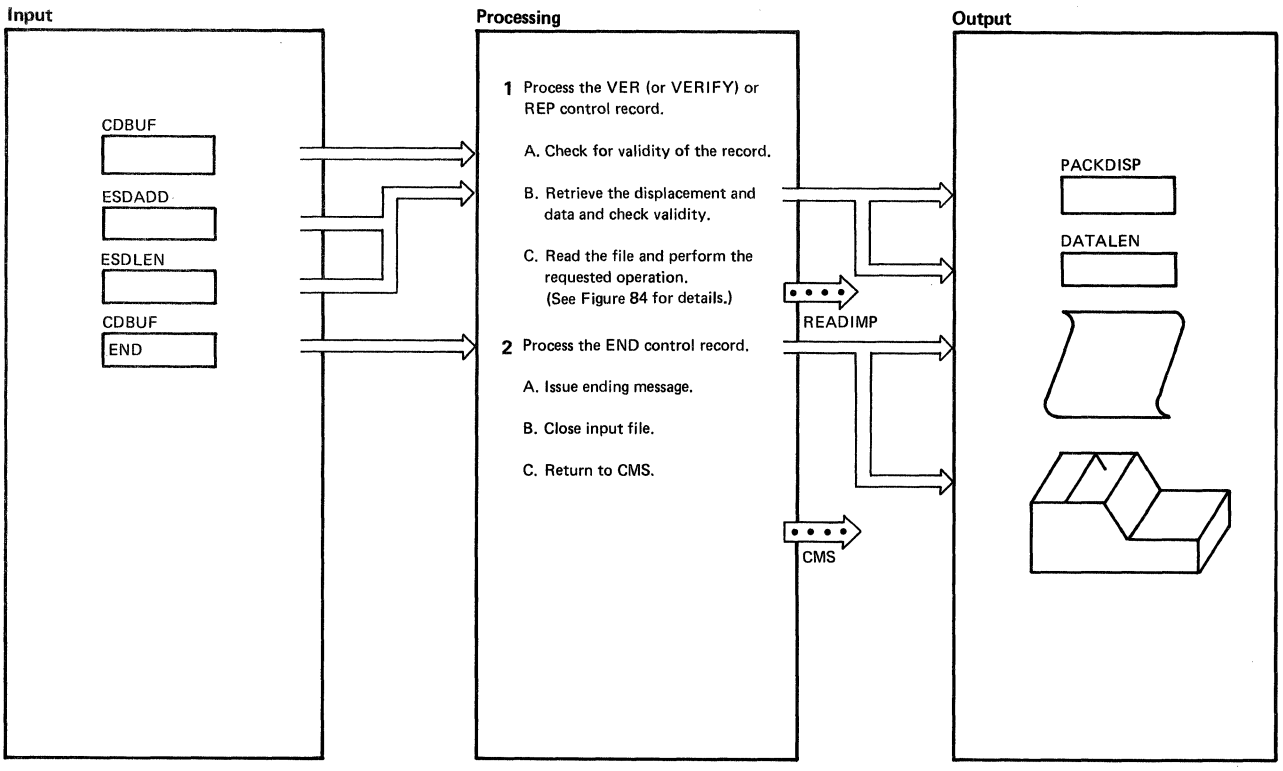
Notes		Module	Label	Ref
1	Retrieve the member name or module name, if specified, from the control record. If an error is encountered, issue the message DMSZAP201W INVALID CONTROL RECORD OR NO GO SWITCH SET Continue by reading another control record.	DMSZAP	DUMPREC DUMPERR	
2	If starting and ending addresses are specified, retrieve them from the control record, check them for validity, and convert them into hexadecimal digits. If either of the addresses is not an even number of digits, issue the message DMSZAP203W - ERROR - OOD NUMBER OF DIGITS - SET NO GO SWITCH and continue by reading another control record.	DMSZAP	DMPNTALL SCANKEY1 DECODE1 PACKVAL INVEREP2	
3	Go to the open routine (PREOPLIB) to locate the member or module and the CSECT desired.	DMSZAP	DMPCSECT PREOPLIB	
4	Use the starting and ending addresses of the CSECT to determine the length of the dump if not otherwise specified. Go to the read text routine to read the file (RDTXT) and then to the print dump routine (PRTDUMP). If all CSECTs are requested, return control to step 3. When the request is satisfied, read another control record (see Figure 78, Step 4).	DMSZAP	STSTART GORDTXT RDTXT PRTDUMP READINP	

Figure 79. DUMP Control Record Processing



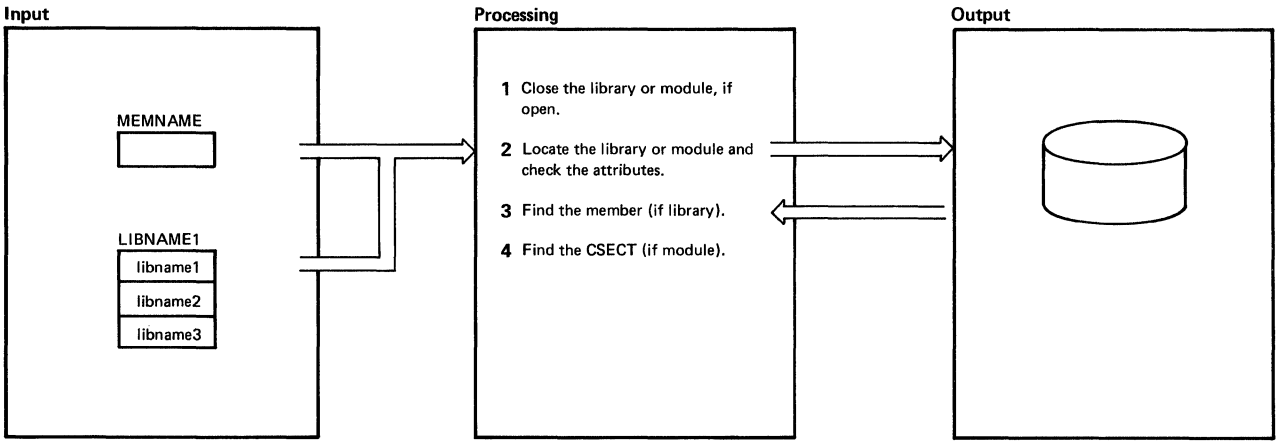
Notes		Module	Label	Ref
1				
	A. Retrieve the member name or module name and the CSECT name, if specified, and check for errors. If errors are found, issue the message	DMSZAP	NAMEREC INVEREP	
	DMSZAP190W Invalid control record or NO GO switch set			
	Continue by reading another control record.			
	B. If no errors are found, open the specified file and locate the desired CSECT. Continue by reading another control record.	DMSZAP	NOCESCT1 PREOPLIB READINP	
2				
	A. Check the NAME control record has been entered. If not, issue the message	DMSZAP	BASEREC INVEREP	
	DMSZAP190W Invalid control record or NO GO switch set			
	Continue by reading another control record.			
	B. Retrieve the BASE address, check it for accuracy, and convert it to hexadecimal.	DMSZAP	CKBASE DECODE1 PACKVAL	
	If the address is not an even number of digits, issue the message		INVEREP2	
	DMSZAP192W Odd number of digits; set NO GO switch			
	and continue by reading another control record.			
	If the file is a MODULE file created with the NOMAP option, accept the BASE address and continue by reading another control record.			
	If the file is a LOADLIB or TXTLIB file, or a MODULE file not created with the NOMAP option, compare the BASE address to the CSECT address. If there is a match, continue by reading another control record.		CKBASE1	
	If the CSECT address is not equal to the BASE address, issue the message		INVEREP2	
	DMSZAP195W Base value invalid; set NO GO switch			
	Continue by reading another control record.			

Figure 80. NAME and BASE Control Record Processing



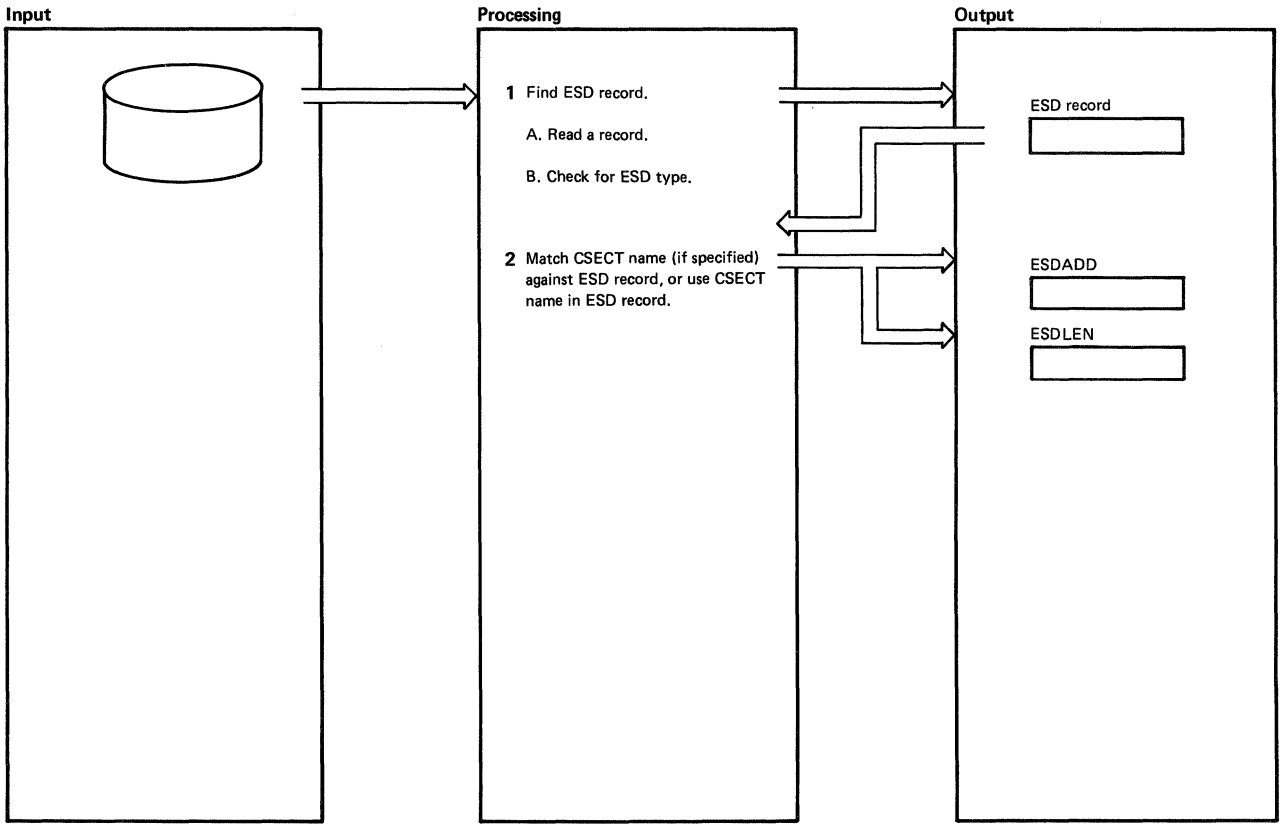
Notes		Module	Label	Ref
1				
A.	If a NAME control record has not been entered or was invalid, issue the message DMSZAP190W Invalid control record or NO GO switch set and return control to READINP to read another control record. Ignore all VER or REP control records until the next NAME control record is encountered.	DMSZAP	GOODTHRE INVEREP	
	If this is a REP control record and the NO GO switch is on, issue the message DMSZAP193W Preceding control record flushed and return control to READINP to read another control record.		INVEREP2	
B.	Check the displacement for validity and convert into hexadecimal digits.	DMSZAP	GOOK SCANKEY1 DECODE1 PACKVAL	
	Retrieve the data field, remove commas from the field, and check that the data are an even number of bytes. If not, issue the message DMSZAP192W Odd number of digits; set NO GO switch and return control to READINP to read another control record.	DMSZAP	SCANKEY1 CKCOMMA2 CKCOMMA3 INVEREP2	
	Convert the data to hexadecimal and add the BASE value to the displacement. Check that the displacement plus the data length will fit within the CSECT. If not, issue the message DMSZAP191W Patch overlaps; set NO GO switch and return control to READINP.	DMSZAP	EQLNTH PACKDAT INVEREP2	
C.	Go to the RDTXT routine to perform the operation, then return control to READINP.	DMSZAP	GOVER RDTXT	
2				
A.	Issue the message DMSZAP750I ZAP processing complete	DMSZAP	COMMEND INVEREP4	
B.	Close the INPUT file, if it is open, and free buffer space.	DMSZAP	CLOSEINP CLRSPCE	
C.	Return to CMS.		NOMORE	

Figure 81. VER/VERIFY or REP and END Control Record Processing



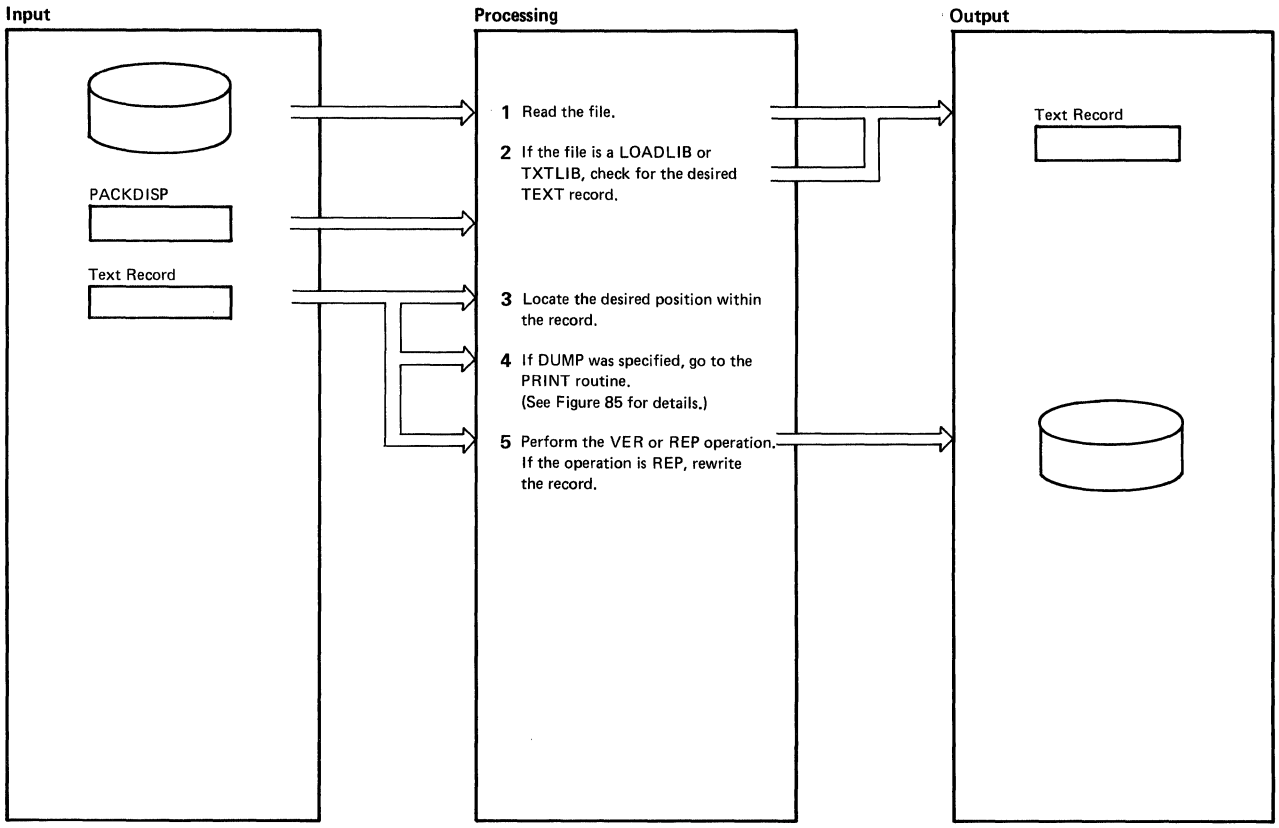
Notes		Module	Label	Ref
1	Close input module and library files, if open.	DMSZAP	PREOPLIB CLOSELIB	
2	If MODULE was specified, locate the module name and search for the module. If the module is found, check the attributes and if they are valid, go to Step 4. Otherwise, issue one of these error messages: DMSZAP210E File *fn ft* is on a read/only disk DMSZAP208E File *fn ft* is not variable record format	DMSZAP	STFDEF LIBRO LIBNTV	
	If the module cannot be found, issue the message DMSZAP002W File *fn ft [fm]* not found and read another control record. Ignore all control records until the next NAME, DUMP, or END control record.		PREOPLB3 PREOPLB5 INVEREP2	
	If LOADLIB or TXTLIB was specified, locate the first library name and search for the member. If the member is found, check the attributes and if they are invalid, issue one of these messages: DMSZAP210E File *fn ft* is on a read/only disk DMSZAP208E File *fn ft* is not variable record format DMSZAP007E File *fn ft fm* is not fixed, 80-character records		STFDEF LIBRO LIBNTV FILENTF	
	Otherwise, go to Step 3 after issuing the message DMSZAP751I Member *membername* found in library *libname*		MEMFND	
	If the library cannot be found, issue the message DMSZAP002W File *fn ft [fm]* not found and locate the next library name and execute Step 2 again. If none of the libraries specified can be found, issue the message DMSZAP002E File [*fn [ft [fm]]*] not found and terminate processing.		PREOPLB3 INVEREP2 LIBNTFD1 NOMORE	
3	When a library is found, read the first record. If the header record or the pointer to the directory is invalid, issue the message DMSZAP056E File *fn ft &lbrkfm]* contains invalid record formats	DMSZAP	OPENFILE PREOPLB4 INVFORM	
	Otherwise, locate the directory record and search for the member name. If the file is a CMS-only (not OS) TXTLIB file and the member name cannot be found, search for the CSECT name. If a member name or CSECT name is found, go to the READCESD routine to find a CSECT record.		READLIB CHKMEM CHKCSECT	
4	If the file is a MODULE, compute the length of the module and its starting and ending addresses. Determine if a map is present and, if not, that no CSECT name was specified, then exit. If a CSECT name was specified, issue the message DMSZAP246W No loader table present for module *fn*; set NO GO switch then exit. If a module map is present, locate the map record and read it. If the map record cannot be found, issue the message DMSZAP056E File *fn ft [fm]* contains invalid record formats	DMSZAP	CHKLDTBL NOTABLE INVEREP2 CHKLDCST INVFORM	
	Otherwise, locate the CSECT specified or the first CSECT in the map, and determine its length, and return control to caller. If the CSECT specified cannot be found, issue the message DMSZAP194W CSECT not found in {member *membername*|module *module*};set NO GO switch and read another control record.		LDRLOOP FINDCLNTH INVEREP2	

Figure 82. Opening the File



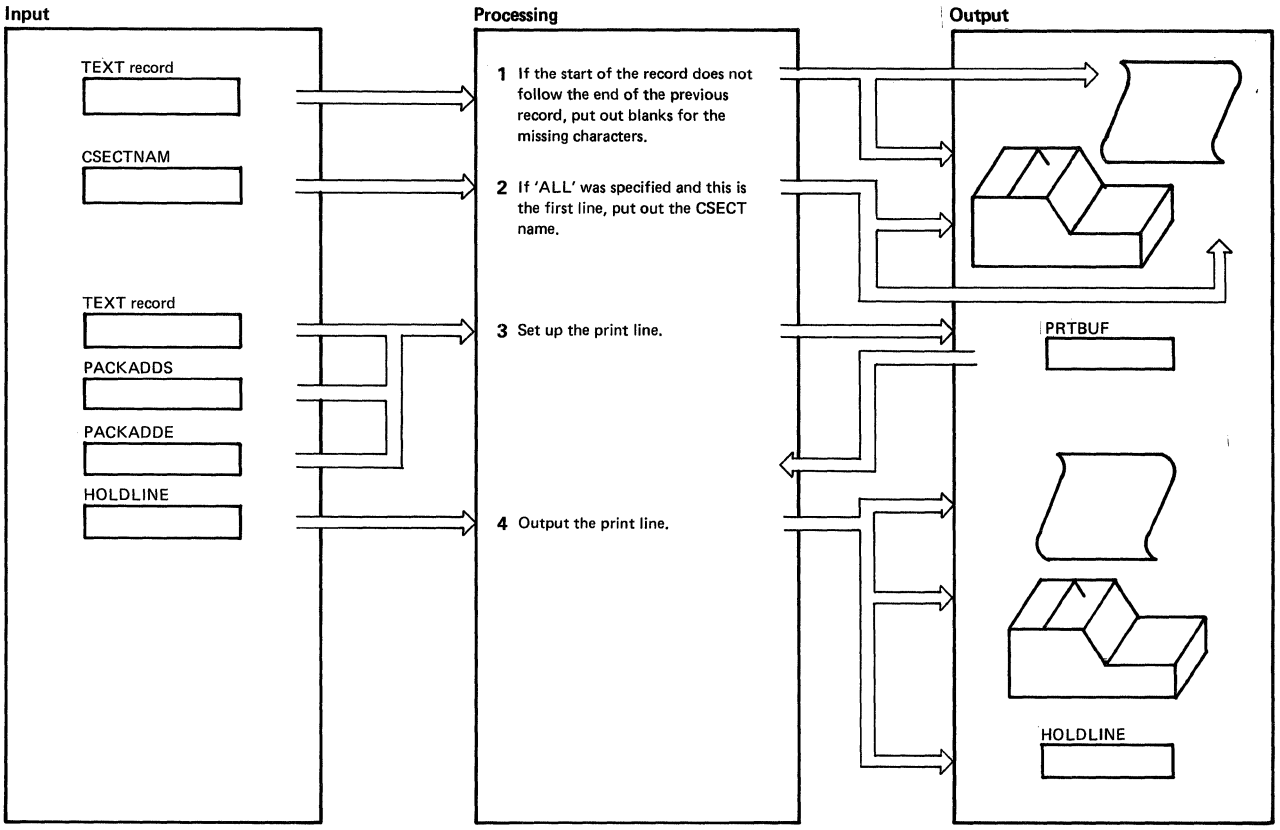
Notes		Module	Label	Ref
1	Read a LOADLIB or TXTLIB member record. Check to see if it is an ESD-type record. If not. re-execute Step 1.	DMSZAP	READCESD TXTEST RDLIB	
2	If a CSECT name was specified in the NAME or DUMP control record, compare it with the CSECT name(s) in the ESD record(s).		SEARCHSD	
	If there is a match, save the starting address and length. If there is not match, issue the message		CSECTFND	
	DMSZAP194W CSECT not found in {member *membername*|module *module*}; set NO GO switch		NOCESD2	
	If no CSECT name was specified in the NAME or DUMP control record, use the first CSECT named in an ESD record.			
	If ALL was specified in a DUMP control record, use the next CSECT name encountered in an ESD record.			
	Control then returns to caller.		MEMEND	

Figure 83. Finding the CSECT



Notes		Module	Label	Ref
1	Read the next record of the file. If the file is a module, go to step 3.	DMSZAP	RDTXT	
2	If the file is a TXTLIB, check for the desired record. If not, repeat step 1. Otherwise, check for valid characters. If there are no valid characters (that is, if the area ia a Define Storage area), and the operation is VER or REP, issue the message DMSZAP248W Invalid VER/REP displacement; set NO GO switch If there are no valid characters, but the operation is DUMP, determine the length of the gap and handle it as a TEXT record.	DMSZAP	RDTXTLIB RDTXTFND	
	If the file is a LOADLIB, check for the desired record. When it is found, check for valid characters as with a TXTLIB and, if valid, read the next record for the actual text.	DMSZAP	RDLDLIB RDLDCHK	
3	Determine the position within the record.	DMSZAP	CHKVER	
4	If the operation desired is DUMP, go to the PRINT routine to print out lines.	DMSZAP	VERCHK PRTDUMP	
5	If the operation is REP, replace each byte read with the data supplied in the REP control record. When the end of the record is reached or the REP operation is completed, rewrite the record.	DMSZAP	VERLOOP VERIFY1 VERIFY2 WRLIB	
	If the operation is VER, compare each byte read with the data in the VER control record. If they do not agree, issue the message DMSZAP200W Verify reject; set NO GO switch		VERLOOP VERIFY1	
	If another record is required, to go step 1. Otherwise, control returns to caller.		RDTXEND	

Figure 84. Reading the Text



Notes		Module	Label	Ref
1	If the start of a new record does not match the end of the previous record, or the requested start of the dump is not found, insert blanks in the output record to represent the bytes not in the file.	DMSZAP	PRTDUMP SETBLANK	
2	If 'ALL' was specified and this is the first line of the CSECT, output the CSECT name.	DMSZAP	NEWLIN PRTHDR	
3	If a line has been started, finish the line. If not, set up the new line. Determine the address of the new line, check that the line does not exceed the requested end of the dump, and move characters from the record into the line. If the line does exceed the requested end of the dump or the record is exhausted, fill the output line as much as possible, convert its characters for printing, and save the pointers. Return control to caller.	DMSZAP	FINLINE NOFSTLN SETADD SETHXLN SETHXLNA CHARCONV PRTDNXT	
4	When the line is ready for printing, convert the non-printing characters to periods, and compare the line to the previous line. If there is a match, save the address of the current line. If there is no match, and addresses have been saved, print the message LINES xxx TO xxx SAME AS ABOVE. Otherwise, print the line and save it in HOLDINE.	DMSZAP	CHARCONV PRTLINE CHKDUP NOTDUP PRTLIN2	

Figure 85. Printing the Dump

Program Organization

This section contains a program description of the DMSZAP module.

DMSZAP

The ZAP service program

Entry Point

DMSZAP -- via the command ZAP

Attributes

Reusable, not disk resident

Entry Conditions

R1	Address of the input parameter list
R15	Address of the entry point

Register Usage

R1	Address of the input parameter list
R2-8	Work registers
R9	Base registers
R10	Link register
R11-12	Base registers
R13	Address of the save area
R14	Return address
R15	Return code

Calls to Other Routines

DMSBRD	To read input disk files
DMSBWR	To write output disk files as a result of REP operation
DMSERR	To handle calls from DMSERR and LINEDIT macros
DMSFNS	To close input and output files
DMSPRT	To handle PRINT command
DMSSMN	To handle OS GETMAIN and FREEMAIN macros

DMSSTT	To provide a copy of an FST
DMSSVT	To process OS macros

External References

None.

Data Areas

File Status Table

Exit Conditions

R15	Return code

Directory

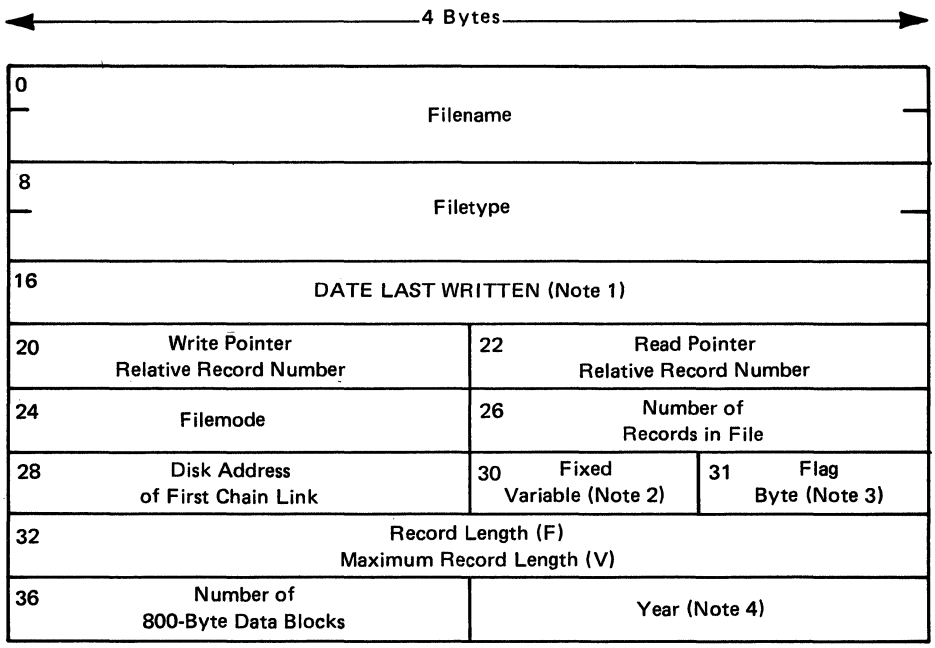
Following is an alphabetical list of the major labels of the ZAP program. The associated method of operation diagrams are indicated and a brief description of the operation performed at the point in the program associated with each label is included.

Label	Figure	Description
BASEREC	80	Processes a BASE control record. Scans for displacement.
CHKLDTBL	82	Locates a CSECT (for a module file) if a name is given
CHKMEM	82	Checks for a member, or, if a CMS TXTLIB, for a CSECT
CLOSELIB	82	Finishes the specified library or module
CLOSINP	81	Closes the input file
CLRSPCE	81	FREEMAINs buffer space
CONEND	81	Processes an END control record
CONSOPT	78	Sets the TERM option
DECODE1	80, 81	Checks that a field is less than six digits
DECODE2	80, 81	Checks that a field is an even number of digits
DMSZAP	77	Saves the input registers and sets addressability
DOWTO		Does a write-to-operator for messages when in terminal mode
DUMPREC	79	Gets the location of the dump and prints it
FDEFINP	78	FILEDEFs the input DCB and opens it
FINDMEM	82	Locates the beginning of a member
FNDCLNTH	82	Locates the boundary of a CSECT
INITOPEN	77	Opens input (if specified) and output (printer) files
INPTOPT	78	Sets the INPUT option
INVEREP	78	Processes the error message for an invalid control record and closes the SYSLIB file
NAMEREC	80	Processes a NAME control record. Scans for the member name and CSECT name.
NAMFOUND	78	Branches to the appropriate routine when a keyword is found in the table
NEWLIN	85	Prints full lines
NOMORE	81	Gets the error code and prior save area address, restores the registers, and returns to DMSITS
NOPRTOPT	78	Sets the NOPRINT option
OPENFILE	81	Opens a library
PREOPLB1	81	Gets the first library name address
PREOPLB4	81	Reads a ZAP file and locates a member (CSECT for a MODULE file if a name was given)
PREOPLIB	82	Opens ZAP files and looks for the library name, if given
PRINTOPT	78	Sets the PRINT option
PRTCARD		Prints a card image
PRTDUMP	85	Prints the requested dump
PRTHDR	85	Prints the name of the CSECT being dumped
PRTLINE	85	Prints a dump line
RDCARD	78	Requests input from the terminal
RDCARD2	78	Reads an input control record file
RDLDLIB	84	Analyzes LOADLIB records
RDLIB	83	Reads the specified library or module

Label	Figure	Description
RDTXT	84	Reads a library searching for the record to be verified or replaced
RDTXTLIB	84	Analyzes TXTLIB records
READCESD	83	Reads a CESD record of a member
READINP	78	Reads a control record from the input file. Writes the control record to the output (SYSPRINT) file. Scans the first keyword from the control record.
SCANKEY1	78	Scans control records
SCANLINE	78	Checks the command line for validity
SEARCHSD	83	Searches a CESD record for an ESD entry with a CSECT name
SETBLANK	85	Spaces over a DS area
STFDEF	82	Issues a STATE for a library file, checks that the disk is in Read/Write mode
TABLOOK	78	Look for a keyword in the table
TXTESD	83	Finds a TXTLIB CSECT
WRCARD	78	Writes a control record and messages to SYSPRINT file
WRLIB	84	Updates the specified library or module

Data Areas

The File Status Table is used by the DMSZAP module:



Notes:

1. Date last written is in packed decimal format MM DD HH MM; for example, 02 20 14 07 represents February 20, 2:07 p.m.
2. F = Fixed-length records. V = Variable-length records.
3. Flag Byte = 0
4. Year is in character form; for example, '72' for 1972

Figure 86. File Status Table Entry

Diagnostic Aids

The ZAP Command Processor (DMSZAP)

Message Code	Label	Figure	Message Text
DMSZAP001E	SCANLINE	78	No filename specified
DMSZAP002E	FDEFINP	78	File [*fn [ft [fm]*]] not found
	PREOPLB3	82	
DMSZAP002W	PREOPLB5	82	File *fn ft [fm]* not found
DMSZAP003E	SCANLINE	78	Invalid option: *option*
DMSZAP007E	FDEFINP	82	File *fn ft fm* is not fixed, 80-character records
	STFDEF		
DMSZAP014E	SCANLINE	78	Invalid function *function*
DMSZAP047E	SCANLINE	78	NO function specified
DMSZAP056E	PREOPLB4	82	File *fn ft* [*fm*] contains invalid record formats
DMSZAP070E	SCANLINE	78	Invalid parameter *parameter*
DMSZAP104S	PREOPLB4	82	Error *nn* reading file *fn ft fm* from disk
	CHKLDTBL	82	
	RDCARD2	78	
	RDLIB	82	
DMSZAP109S			Virtual storage capacity exceeded
DMSZAP190W	INVEREP	78	Invalid control record or NO GO switch set
		79	
		80	
		81	
DMSZAP191W	DUMPREC	82	Patch overlaps; set NO GO switch
	GOODTHRE		
DMSZAP192W	DECODE1	79	Odd number of digits; set NO GO switch
		81	
	GOODTHRE	80	
DMSZAP193W	GOODTHRE	82	Preceding control record flushed
DMSZAP194W	OPENFILE	82	CSECT not found in {member *membername*|module
	READCESD	83	*module*}; set NO GO switch
DMSZAP195W			BASE value invalid; set NO GO switch
DMSZAP200W	VERIFY1	84	Verify reject; set NO GO switch
DMSZAP208E	STFDEF	82	File *fn ft* is not variable record format
DMSZAP210E	STFDEF	82	File *fn ft* is on a read/only disk
DMSZAP245S	WRCARD	78	Error *nnn* on printer
DMSZAP246W	CHKLDTBL	82	No loader table present for module *fn*; set NO GO switch
DMSZAP247W	PREOPLB3	82	Member *membername* not found; set NO GO switch
DMSZAP248W	RDTXTLIB	84	Invalid VER/REP displacement; set NO GO switch
	RDLDLIB	83	
DMSZAP249I			Dummy log entry in file *fn* ZAPLOG *fm*
DMSZAP750I	CONEND	81	ZAP processing complete
DMSZAP751I	OPENFILE	82	Member *membername* found in library *libname*

Index

Chapter 9. EREP/Error Recording Interface

Introduction

The VM/SP method of editing error records accumulated on the VM/SP error recording area or stored on other devices makes use of the OS/VS EREP Edit and Print programs. To use these programs from a VM/SP virtual machine environment requires the use of the DMSIFC module which is called by DMSITS when the CPEREP (EXEC) command is processed.

DMSIFC loads DMSREA and several modules of OS/VS EREP into main storage and then passes control to OS/VS EREP.

Prior to passing control to EREP, DMSIFC does the following:

- Issues FILEDEFs for files needed by OS/VS EREP.
- Reads control parameters from the user and puts them into an OS-compatible parameter (PARM) list format to be passed to OS/VS EREP.
- Creates a SYSIN file of control parameters from the control parameters that have been entered.
- Uses the HNDSVC macro instruction to prepare for trapping the EXCPs (SVC 0) that OS/VS EREP will issue when it attempts to read records from the SY1.LOGREC data set.

NOTE: HNDSVC is also used to prepare to trap BLDLs (SVC 18) that OS/VS EREP will issue.

The several modules of OS/VS EREP that must be loaded by DMSIFC are those that contain VCONs or that are needed in the process of resolving VCONs. DMSIFC invokes the CMS INCLUDE command dynamically to load these OS/VS EREP modules from CPEREP's two TXTLIB files. Other modules of OS/VS EREP that do not contain VCONs are loaded later (from the two TXTLIB files) by OS/VS EREP itself as they are needed.

DMSIFC passes control to OS/VS EREP by executing an OS LINK (to EREP's IFCEREP1 module, which has already been loaded). The OS-compatible parameter list built by DMSIFC is passed to IFCEREP1 at this time and OS/VS EREP begins to execute.

EREP issues set EXCPs for I/O to the OS SYS1.LOGREC data, which are intercepted by CMS. CMS transfers control back to DMSIFC, which simulates the EXCPs so that they appear to access a SYS1.LOGREC data set. This simulation results in calls to DMSREA to supply records contained in the VM/SP error recording area.

EREP issues BLDLs (SVC 18) to determine whether or not EREP modules needed for certain error records are present in the TXTLIBs. The standard

CMS simulation of OS BLDL does not include the JOBLIB/STEPLIB form of BLDL which EREP uses here. Therefore, these BLDLs are intercepted and are simulated by DMSIFC.

When EREP is finished executing, it exits (returns to DMSIFC which invoked it). Before returning to CMS, DMSIFC does some cleaning up. Temporary files are erased and FILEDEFs issued by DMSIFC are cleared with the following exceptions: the EREPPT, ACCIN, TOURIST, and ACCDEV FILEDEFs are not cleared because they may have been entered by the user or by DMSIFC but DMSIFC has no way of knowing which. Since they should not be cleared if they were entered by the user, DMSIFC never clears them.

Method of Operation

This section describes the VM/SP interface between CMS (the Conversational Monitor System) and the OS/VS EREP program. Figures 88 and 89 describe the functions of the interface modules and serve as a guide to the program listings. The labels shown indicate the closest, nonmacro expansion label to the function being documented. These diagrams are not terribly detailed, therefore, some functions are not shown. Use the Directory and Program Organization section to find the labels in the program listings for any routines that are not shown in the Method of Operation section. Figure 87 shows the relationship of these figures.

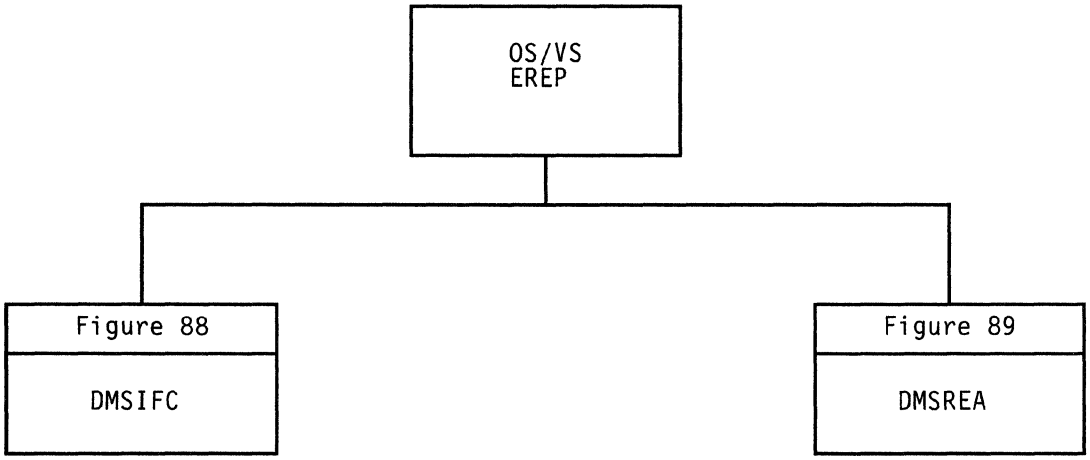
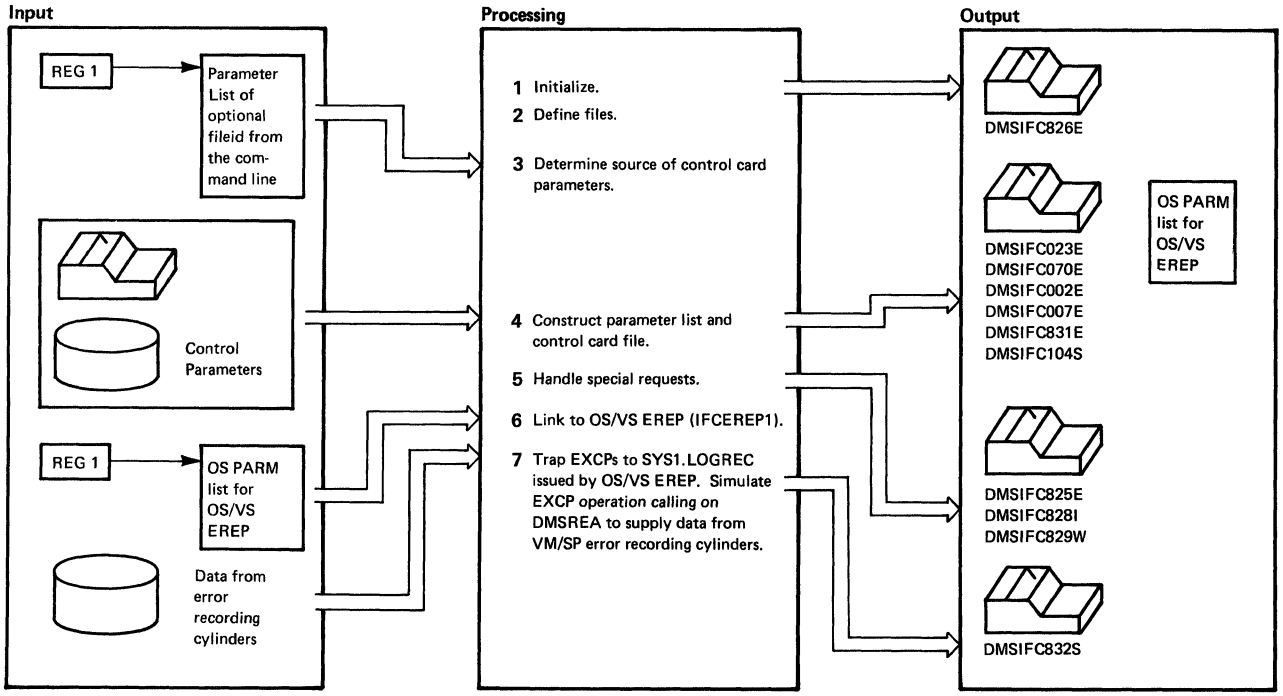


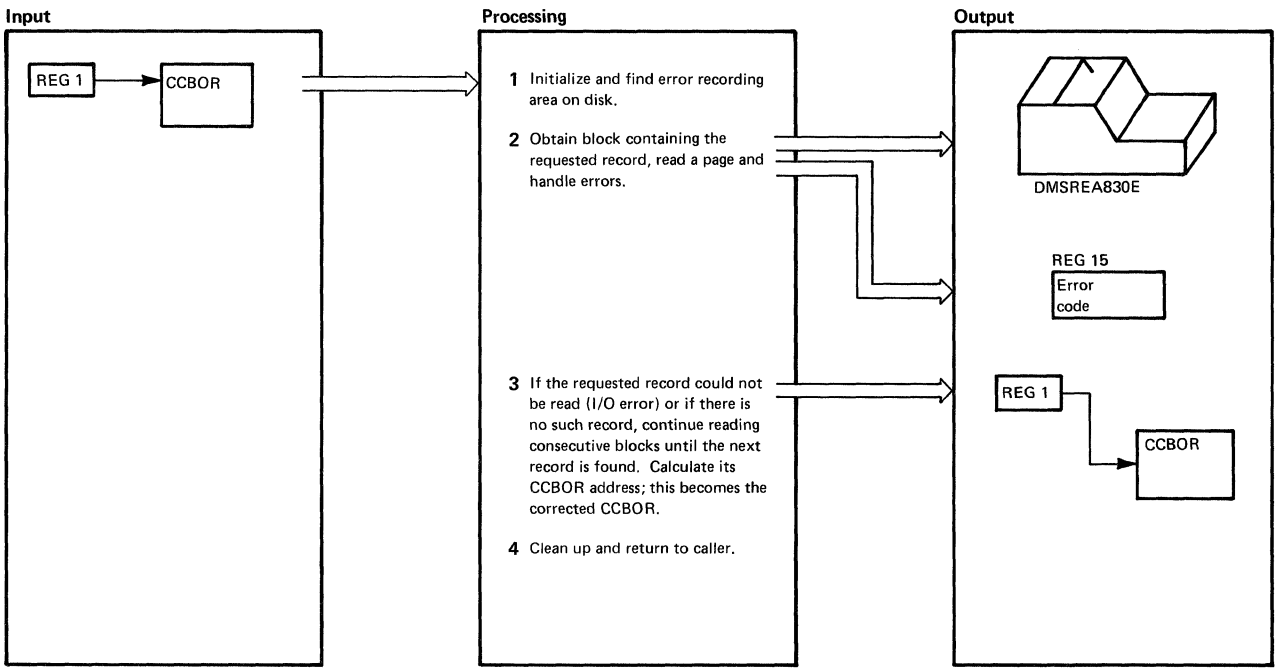
Figure 87. Key to EREP/Error Recording Interface Method of Operation Figures



Notes		Module	Label	Ref
1	The initialization procedures include: ● Standard linkage and addressability functions. ● Loading and resolving VCONs in OS/VS EREP decks. ● Loading DMSREA. ● Turning off flag in CMS nucleus to cause OS simulation. ● Setting COMPSWT in CMS nucleus to load LINK and LOAD macros to be entered in TEXT files. ● Establishing handling of SVC 76, SVD 18, and SVC 0.	DMSIFC	DMSIFC	Figure 88
2	Invoke FILEDEF to define: ● Printer file (EREPPT). ● SYSIN file (SYSIN). ● Dummy file for SYS1.LOGREC (SERLOG). ● Error file (TOURIST). ● Work file (DIRECTWK). ● Accumulation tape file (ACCDEV). ● History input tape (ACCIN).		NORWDISK RDYACC RDYHIST OPER12 HAVETYPE NOEXTRA	
3	Determine where control parameters are to be taken from (Control file or terminal).		BADATTR GOODATTR	
4	Set up to read parameters. Obtain storage for OS PARM list to be passed to EREP. Read control parameters, generating the OS PARM list and a SYSIN file as output. Call subroutine to read control parameters. Handle errors.		PARMWORK RDERR1 PLISTBLD	Figure 88
5	If CLEAR is specified with other parameters, type an error message. If CLEAR is specified properly, call subroutine to erase error records from the VM/SP error recording cylinders. Subroutines handle each parameter information. CLEARF parameter (determines validity by examining processor identity. If not 3031, 3032, or 3033 processor reject command but if valid, erase error records from the error records from the error recording cylinders, then initialize SRF frames to the beginning of the error recording cylinders.) ● CLEAR parameter ● TERMINAL parameter (stops reading from control file on disk and goes to terminal to read additional control parameters) ● SHARE control statement ● ACC parameter ● HIST parameter ● MERGE parameter ● THRESHOLD parameter ● ZERO parameter ● DASDID control statement ● LIMIT control statement ● CONTROLLER control statement.		WANTCLR CLEARRTIN HCLEARF HCLEAR HTERM HSHARE HACC HHIST HMERGE HTHRES HZERO HDASDID HLIMIT HCONTROL	
6	Load the address of the word that points to the OS PARM list built for OS/VS EREP and LINK to IFCEREP1.			
7	EXCP SVCs from EREP are intercepted and simulated so they appear to access a SYS1.LOGREC data set. Simulation causes calls to DMSREA for VM/SP error records. BLDL SVCs from EREP are also trapped and simulated by DMSIFC.		DMSIFC0 DMSIF18	

Figure 88. DMSIFC

EREP/Error Recording Interface



Notes		Module	Label	Ref
1	The initialization procedures include: ● Saving registers. ● Setting return code to zero. ● Issuing DIAGNOSE X'2C' to locate beginning of error recording area and number of cylinders. ● Setting and checking "first time" switch. ● Checking CCBOR address passed for validity. *Note:* A CCBOR disk address is a disk addressing format devised solely for use in CPEREP and resembles the commonly used CCHHR disk address. In a CCBOR address the fields have the following meaning: **Field** **Meaning** CC Relative cylinder within the VM/SP error recording area, for example: CC= X'0000' for the first cylinder of the error recording area. = X'0001' for the second cylinder of the error recording area. B The number of the desired 4K block within the cylinder. The first 4K block in a cylinder is X'01'. O Zero R The number of the desired record within the 4K block. The first record in a block is X'01'.	DMSREA	DMSREA FIRSTSW OPER4	Figure 89
2	DMSREA converts the CCBOR address to a VM/SP Control Program Internal Format address and issues a DIAGNOSE X'30' to read the block into the buffer. If the requested block is found, return to caller. If specified cylinder is outside error recording area, sets error code in register 15 for invalid cylinder. If end of cylinder and no more cylinders are available, sets register 0 or zero, indicating end-of-file to caller; otherwise, advance to next cylinder. If an I/O error occurs so that the block could not be read, issue message DMSREA830E.		OPER5 OPER7 OPER16 OPER17 OPER7 OPER9	
3	If requested record was not found, read next block and return first record from this block. If block is empty or unreadable, continue reading blocks until a record is found or until end-of-file is reached. Use CCBOR address of the record found as the corrected CCBOR value to be returned to the caller. Make register 1 point to the CCBOR address. *Note:* The CCBOR record addresses are passed back to OS/VS EREP (as a result of the EXCP simulation) as if they were CCHHR addresses. EREP never notices the difference and, as a result, EREP uses CCBOR addresses in all its I/O operations to the SYS1.LOGREC data set.		OPER10	
4	Restore registers (except output parameter registers) and return to caller.		OPER15	

Figure 89. DMSREA

Program Organization

This section includes program descriptions of modules DMSIFC and DMSREA.

DMSIFC

Allows virtual users to edit and print VM/SP error recordings under CMS via the OS/VS EREP Edit and Print Program (IFCEREP1).

Entry Point

DMSIFC

Routines Called

- IFCEREP1 via LINK to edit and print VM/SP error recording area
- DMSREA via BALR to read a specified record from the VM/SP error recording area
- DMSLAD via BALR to determine which read/write disk has the most space
- DMKIOG via DIAGNOSE to clear requested recording area
- STATE/STATEW via SVC to perform CMS functions
- ERASE via SVC to perform CMS functions
- INCLUDE via SVC to perform CMS functions.

Attributes

Nonreusable, CMS User Area, and called by CMS

Registers at Entry

Reg	Use
R1	CMS parameter list address
R13	Save area address
R14	Return address

Registers at Exit

Reg	Use
R0-R14	Restored
R15	One of the following return codes:

Return Code	Meaning
12	CLEAR specified with other parameters
24	An invalid parameter or no filetype was specified
28	The file was not found
32	The file was not a fixed-length format
56	GLOBAL command was not issued for CPEREP's TXTLIBs
60	An I/O error caused one or more of the 4K blocks of error records to be skipped
62	More than the maximum number of characters in options specified
88	Attempt to set to zero was suppressed Requires privilege class F
100	Error reading file from disk

Register Usage

Reg	Use
R0-R1	Parameter registers
R2-R9	Scratch
R10-R11	Spares, not used
R12	Base register
R14-R15	Link registers

External References

Area	Purpose
CURRSAVE	Contains address of the current system save area when control is received to handle an SVC as requested by the HNDSVC macro
OSSFLAGS	OS simulation flags in the NUCON area
DOSFLAGS	DOS simulation flags in the NUCON area
AADTLKW	Contains address of routine that determines which read/write disk has the most space (In the NUCON)

Area	Purpose
TXTLIBS	Indicates whether or not any TXTLIBs have been globaled (In the NUCON)
TXTDIRC	Indicates whether or not any TXTLIBs have been globaled. (In the NUCON; points to the first directory in the chain of global TXTLIB directories)

The functions performed by DMSIFC can be summarized as follows:

1. Performs standard linkage and addressability functions

2. Invokes CMS LOAD function to load and resolve VCONs in about a dozen EREP object decks. NOTE: All other EREP object decks are brought into storage later, as needed, by OS LOAD and LINK macros issued by OS/VS EREP.

3. Invokes STRINIT function. Indicates that area above presently loaded programs is the beginning of free storage.

4. Turns off the DOSSVC flag in the CMS nucleus so that OS simulation can be used. Sets COMPSWT in CMS nucleus so that OS LOAD and LINK macros bring in TEXT files rather than module files. Invokes OS LOAD to load DMSREA into storage and saves its address so it can be called later during the EXCP simulation.

5. Establishes handling of SVC 76 (error log), SVC 18 (BLDL), and SVC 0 (EXCP).

6. Invokes FILEDEF function to define:

   Printer file for EREP
   SYSIN file to be created for EREP
   Dummy file for EREP to open and close as SYS1.LOGREC
   "TOURIST" error file to the terminal
   DIRECTWK work file on disk

7. Gets the command line arguments and determines if a control file is provided for input. If so, sets up to read parameters from the control file, otherwise, sets up to read parameters from the terminal.

8. Issues a DMSFREE macro to get storage for building OS parameter list to be passed to EREP.

9. Gets input parameters (from control file or terminal) and constructs equivalent OS/VS EREP parameter list and SYSIN control card file.

10. If CLEAR was specified, and it was not the only parameter specified, types an error message to the terminal and does housekeeping and exits to CMS.

11. If CLEAR was specified correctly, calls a subroutine to issue the DIAGNOSE that clears the appropriate records from the VM/SP error

area, then does housekeeping and exits to CMS. If CLEARF was specified, read CPU and director frames from SRF device and write on error area.

12. Invokes FILEDEF to define the accumulation tape file if requested. Issues the tape control macros necessary to position tape for subsequent write operations.

13. Invokes FILEDEF to define history input tape if requested and makes sure that it is rewound.

14. Links to OS/VS EREP (IFCEREP1).

15. Simulates BLDL SVCs issued from OS/VS EREP. Simulates EXEC SVCs issued from OS/VS EREP so they will appear to access a SYS1.LOGREC data set. EXCP simulation will result in calls to DMSREA to get records from VM/SP error recording area.

16. Eventually OS/VS EREP is done and control returns from that LINK done above.

17. Housekeeps all indicators and switches, frees any storage obtained for the OS parameter list area, clears handling of SVC 0, SVC 18, and SVC 76; and clears any FILEDEFs that were set up by CPEREP.

18. Exits to CMS.

DMSREA

Reads a specified logical record from the VM/SP error recording area and returns it to the caller.

Entry Point

DMSREA

Routines Called

- DIAGNOSE X'2C' to find the beginning of the recording area on the system disk, and the size of the error recording area.
- DIAGNOSE X'30' to read a page size record from the error recording area.
- DMSERR via macro SVC to write error messages to the console.

Attributes

Nonreusable, CMS User Area, enter via CALL.

Registers at Entry

Reg	Use
R1	Address of CCB0R DASD record address
R13	Save area address
R14	Return address

Registers at Exit

Reg	Use
R0	Nonzero: address of variable-length record being returned. The first 4 bytes are the record descriptor word containing the record length. Zero: end-of-file; no record was at or beyond the entered address.
R1	Address of CCB0R DASD record address (sometimes corrected).
R13	Save area address.
R15	One of the following return codes:

Return Code	Meaning
00	Nothing unusual
04	Empty 4K block skipped
08	Invalid CC value in CCB0R address that was entered
60	I/O error accompanied by message DMSIFC830E

Register Usage

Reg	Use
R0-R9	Scratch
R10-R11	Spares, not used
R12	Base
R13	Save area address
R14-R15	Scratch

External References

None. The functions performed by DMSREA can be summarized as follows:

1. Issues the DIAGNOSE command to find the beginning of the VM/SP error recording area and the size of the area.
2. Reads a requested record from the VM/370 error recording area.
3. Returns the next logical record to the caller when the requested record does not exist or cannot be read and revises the caller's specified CCB0R address accordingly.
4. Handles errors.

Directory

Following is an alphabetical list of the major labels of modules DMSIFC and DMSREA. The associated method of operation diagrams are indicated and a brief description of the operation performed at the point in the program associated with each label is included.

Label	Figure	Description
BADATTR	88	Handles file not fixed
CLEARRTN	88	Logically erases VM/SP error recording area
DMSIFC0	88	Handles trapped EXCPs issued by EREP
DMSIFC18	88	Handles trapped OS BLDL macros issued by EREP
OPER7	89	Issues I/O error reading records message
OPER9XX	88	Handles specification of CLEAR when entered with other parameters
NOEXTRA	88	Handles file not found
EXIT0	88	Restores registers for exit from DMSIFC
EXIT1	88	Clears handling of SVCs
EXIT3	88	Frees storage allocated for OS parameter list
EXIT9	88	Frees storage allocated for SVC simulation
FIRSTSW	89	Sets indication of first time DMSREA is called
HACC	88	Directs addition of ACC parameter to OS parameter list being built for EREP
HAVETYPE	88	Handles the specification of an extra parameter on the CPEREP command line
HCLEAR	88	Clears all error records from the error recording area
HCLEARF	88	Clears SRF frame records and all error records and reformats the error recording area
HCTLCRD	88	Writes CTLCRD information into SYSIN file for EREP to read
HHIST	88	Directs addition of HIST parameter to OS parameter list being built for EREP
HMERGE	88	Directs addition of MERGE parameter to OS parameter list being built for EREP
HMES	88	Directs addition of MES and THRESHOLD parameters to OS parameter list being built for EREP
HRDESUM	88	Directs addition of RDESUM parameter to OS parameter list being built for EREP
HSHARE	88	Writes SHARE parameter into SYSIN file for EREP to read
HZERO	88	Directs addition of ZERO parameter to OS parameter list being built for EREP
OPER4	89	Checks CC portion of entered CCB0R for valid range
OPER7	89	Prepares for and issues DIAGNOSE command to read a page of error records
OPER9	89	Prepares to read first record of next block
OPER10	89	Retains address of block just read into buffer. Decides whether this block contains data or is empty
OPER12	88	Handles special considerations for ACC parameter specification
OPER13	88	Handles special considerations for HIST parameter specification
OPER15	89	Restores registers and returns to caller from DMSREA

Label	Figure	Description
OPER16	89	Sets error code for invalid cylinder
OPER17	89	Handles end of cylinder indication
PARMWORK	88	Issues DMSFREE macro to get storage for building OS parameter list
PLISTBLD	88	Adds passed parameters to OS parameter list being built for EREP
RECLOOP	88	Increments counters to step through buffer until empty or end of specified record found
RDCTLINE	88	Reads and returns one line of control parameters from the terminal or control file
RDERR1	88	Handles errors reading control file from disk
WANTCLR	88	Handles calling subroutine to perform CLEAR

Data Areas

DMSREA

No system data areas are used by DMSREA. However, DMSREA uses 4K of unallocated storage at absolute location X'21000' as a page buffer in which to read the 4K blocks of error records.

DMSIFC

DMSIFC uses ADTECT (the ADT macro) and FSTSECT (FSTB macro) to read from but does not store into them. It uses SSAVE and NUCON also. SSAVE is the CMS system save area that saves the value of the SVC old PSW, the caller's registers, and other necessary control information required to process SVCs and return to the caller. NUCON contains all the nucleus constants for CMS. These are either listed at the end of the module or a description can be found in the *VM/SP Data Areas and Control Block Logic Volume 2 (CMS).*

Diagnostic Aids

Message Code	Label	Figure	Message Text
DMSIFC002E	NOEXTRA	88	[INPUT/OVERLAY] {FILE[(S)]/DATA SET ['fn[ft[fm]],]' NOT FOUND
DMSIFC007E	BADATTR	88	FILE 'fn ft fm' [IS] NOT FIXED. 80 CHAR. RECORDS
DMSIFC023E	NORWDISK	88	NO FILETYPE SPECIFIED
DMSIFC070E	HAVETYPE	88	INVALID {PARAMETER 'parameter'/ARGUMENT 'argument'
DMSIFC104S	RDERR1	88	ERROR 'nn' READING FILE 'fn ft fm' FROM DISK
DMSIFC825E	OPER9XX	88	'CLEAR' IS VALID ONLY WHEN SPECIFIED BY ITSELF
DMSIFC826E	DMSIFC	88	EREP TXTLIBS NOT FOUND
DMSIFC828I	CLROKAY	88	CPEREP ZERO OR CLEAR HAS BEEN COMPLETED
DMSIFC829W	CLEARRTN	88	ATTEMPTED 'ZERO' WAS SUPPRESSED. REQUIRES PRIVILEGE CLASS F
DMSIFC831E	PLISTBLD	88	MORE THAN 100 CHARS. OF OPTIONS SPECIFIED
DMSIFC832S	EXGENERR	88	SOFTWARE INCOMPATIBILITY AT THE CPEREP-EREP INTERFACE. CODE=nnn
DMSREA830E	OPER7	89	I/O ERROR READING A RECORD FROM THE ERROR RECORDING CYLINDERS

Index

Chapter 10. MSS Communicator

Introduction

The DMKMSS program operates under the control of either OS/VS1 or OS/VS2 (MVS) in a virtual machine. It is a communications interface between the VM/SP control program and the MSS Mass Storage Control. It uses a combination of CP-generated attention interrupts on a virtual I/O device, the DIAGNOSE code X'78' instruction, and OS/VS SVC 126 to provide communications.

Requests are received from VM/SP in response to a DIAGNOSE code X'78' instruction issued by DMKMSS. They are passed to the MSC using the standard OS/VS SVC 126. Responses are received from the MSC and returned to VM/SP using diagnose.

Method of Operation

This section describes the two major sections of the DMKMSS program.

Figure 91 on page 238 shows initialization using OS/VS control blocks.

Figure 92 on page 239 shows the processing of a VM/SP request.

Figure 90 shows the relationship of these figures.

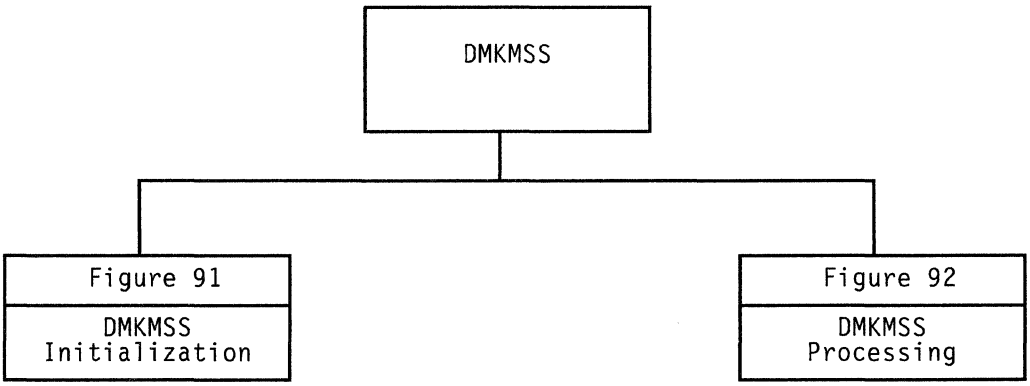
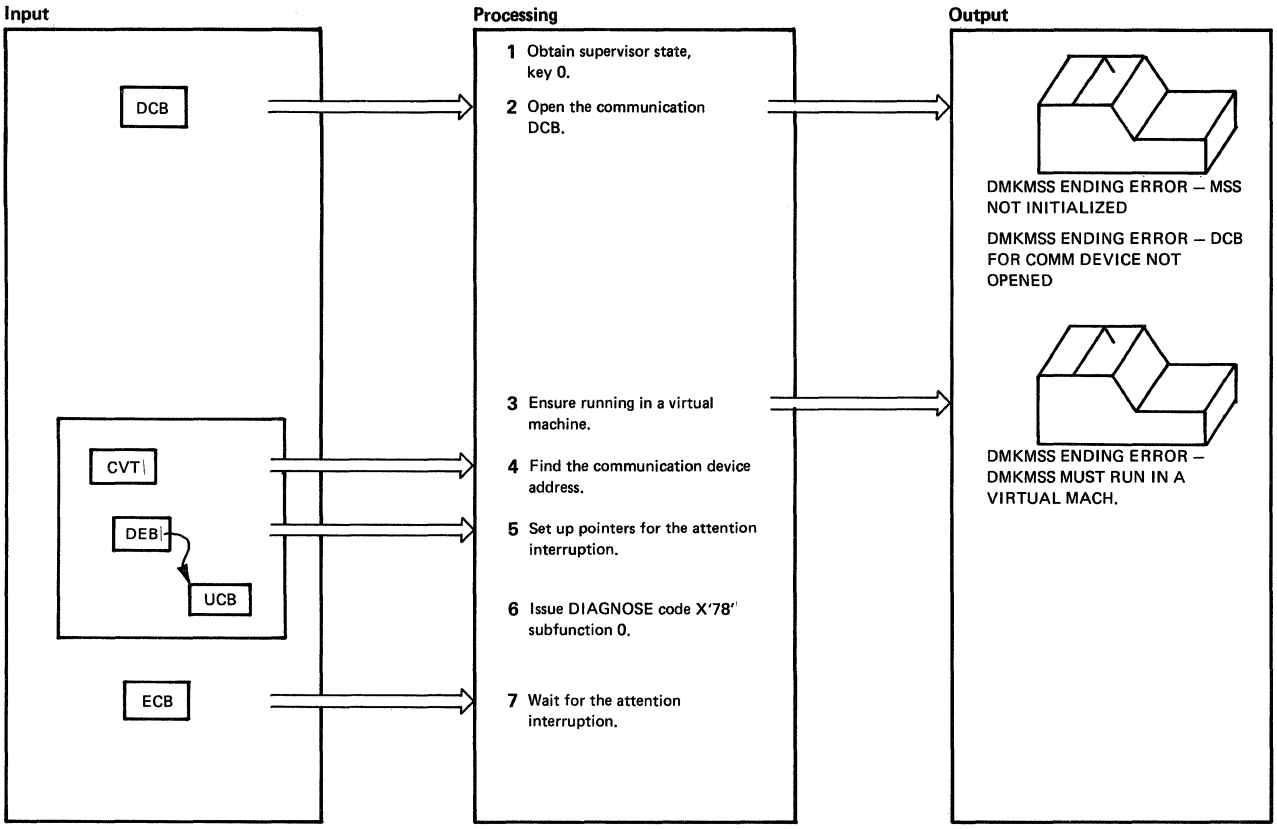
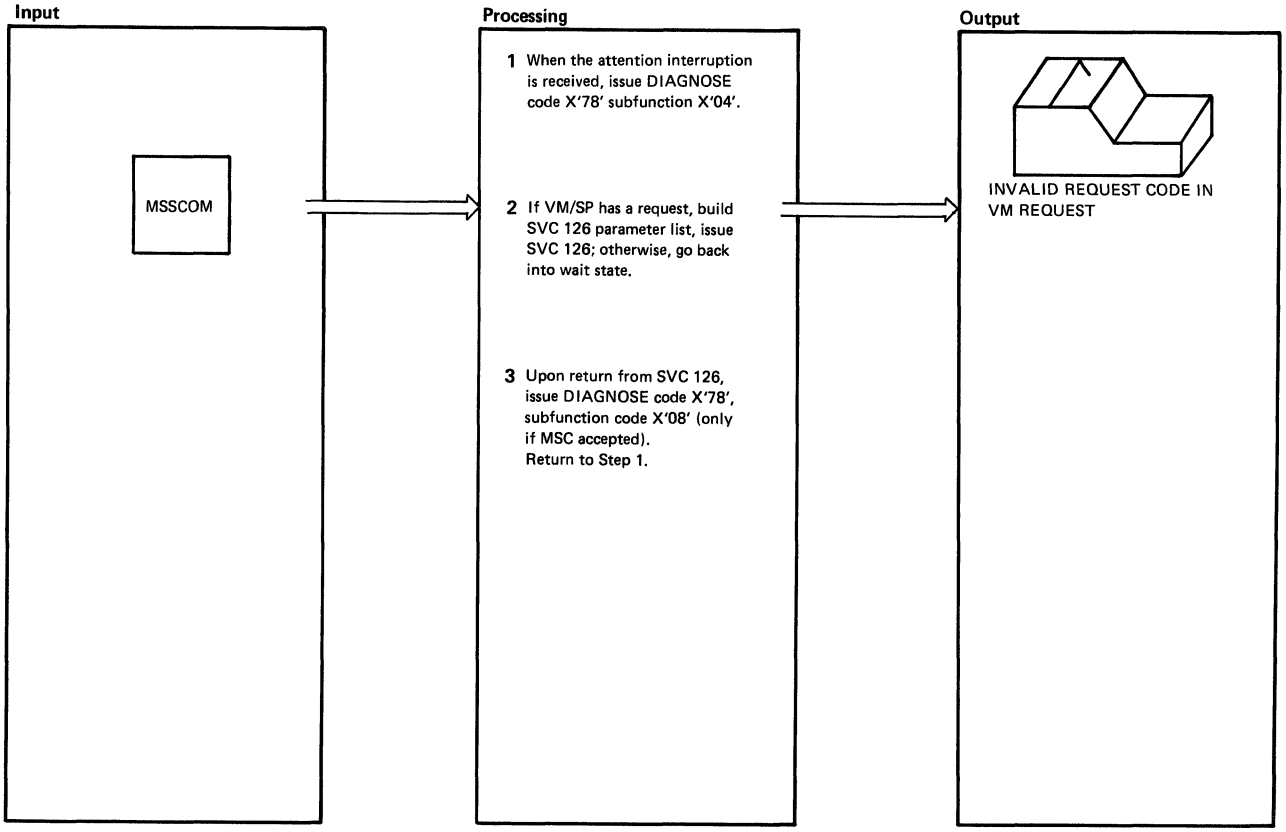


Figure 90. Key to the DMKMSS Method of Operation Figures



Notes		Module	Label	Ref
1	Use the VS MODESET SVC to get into supervisor state, key 0.	DMKMSS		
2	Use the VS OPEN SVC to connect the DCB to the VS control block. If MSS initializes incorrectly, issue message. If the DCB for the communication device does not open, issue message.		RF00092 RF00182	
3	Use the STIDP instruction to ensure running in a virtual machine. If not running in a virtual machine, issue message.		RF00082 RF00190	
4	Follow pointers through the DCB, DEB, and UCB control blocks to find the channel/unit address assigned by the VS scheduler.		L1	
5	Set the MSC's attention table index in the communication device's UCB. Also store the address of the ECB to be waited on in an unused field of this same communication UCB.			
6	Build and issue the DIAGNOSE code X'78' instruction to tell VM/SP the channel/unit address.		PROLOG	
7	Issue VS WAIT SVC, specifying that the event control block will be posted when the attention interruption is received.			

Figure 91. DMKMSS Initialization



Notes		Module	Label	Ref
1	This loop will run in the VS machine as long as MSS support is in effect. The DIAGNOSE X'78' instruction points to a buffer in DMKMSS into which VM/SP places an MSSCOM, or zeros.	DMKMSS	MAINLOOP	
2	Look at MSSCOM to determine volume serial, 3330V device address, and type of request (mount or demount). If the request is invalid, issue a message. If there are no outstanding requests, go into a wait state.		L2 RF00149 RF00122	
3	The SVD 126 routines issue orders to the MSC. If the MSC rejects the order, it sends a unit check as ending status. SVC then sets a non-zero return code in register 15.		DIAG MSSCHECK	

Figure 92. DMKMSS Processing

Program Organization

This section describes the program organization of the DMKMSS module.

DMKMSS

The MSS communicator program

Attributes

Reentrant

Entry Point

DMKMSS

Register Usage

Reg	Use
R0-R9	Work registers
R10	Workarea base
R11	Program base
R12	Work register
R13	Register savearea base
R14-R15	Work registers

Directory

Following is an alphabetical list of the major labels in the DMKMSS program. It indicates the associated method of operation diagrams and it provides a brief description of the operation performed at the point in the program associated with each label.

Label	Figure	Description
DIAG	92	Issues DIAGNOSE code X '78' subfunction X'08' or X'0C'.
L1	91	Follows pointers through the DCB, DEB, and UCB to find the communicator device address
L2	92	Determines the type of MSS request (mount or demount)
MAINLOOP	92	Issues DIAGNOSE code X'78' subfunction X'04', requesting work
MSSCHECK	92	Sets the MSC completion code for VM/SP
PROLOG	91	Initializes for DIAGNOSE code X'78' subfunction X'00'
RF00082	91	Issues STIDP instruction to ensure running in a virtual machine.
RF00092	91	Issues message that MSS is not initialized
RF00122	92	Waits for the communicator device attention interruption
RF00149	92	Issues message for invalid request code in VM request
RF00182	91	Issues message that DCB is not opened
RF00190	91	Issues message that this must run in a virtual machine

Data Areas

The OS/VS control blocks used (CVT, DCB, DEB, and UCB are described in *OS/VS1 System Data Areas* and in *OS/VS2 System Programming Library: Debugging Handbook Volume 2* .

The MSS communicator control block (MSSCOM) is described in *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* .

Diagnostic Aids

Following is a list of the messages issued by the DMKMSS program. The nearest label and the associated method of operation figure are identified.

Label	Figure	Message Text
DUMPWRT		
RF00092	91	DMKMSS ENDING ERROR - MSS NOT INITIALIZED
RF00149	92	INVALID REQUEST CODE IN VM REQUEST
RF00182	91	DMKMSS ENDING ERROR - DCB FOR COMM. DEVICE NOT OPENED
RF00190	91	DMKMSS ENDING ERROR - DMKMSS MUST RUN IN A VIRTUAL MACH.

Index

Chapter 11. 3800 Utility Programs

Introduction

The GENIMAGE, IMAGELIB, and IMAGEMOD commands allow an installation to maintain 3800 printer character sets and image libraries. The GENIMAGE command (DMKIMG) creates character arrangement tables, library character sets, graphic modification modules, copy modification modules, and forms control buffers. The IMAGELIB and IMAGEMOD commands load these modules into an image library.

DMKIMG

DMKIMG, invoked by the GENIMAGE CMS command, uses the IEBIMAGE program to create TEXT images that will be used by the 3800 Printer Model 1 or Model 3.

DMKNMT

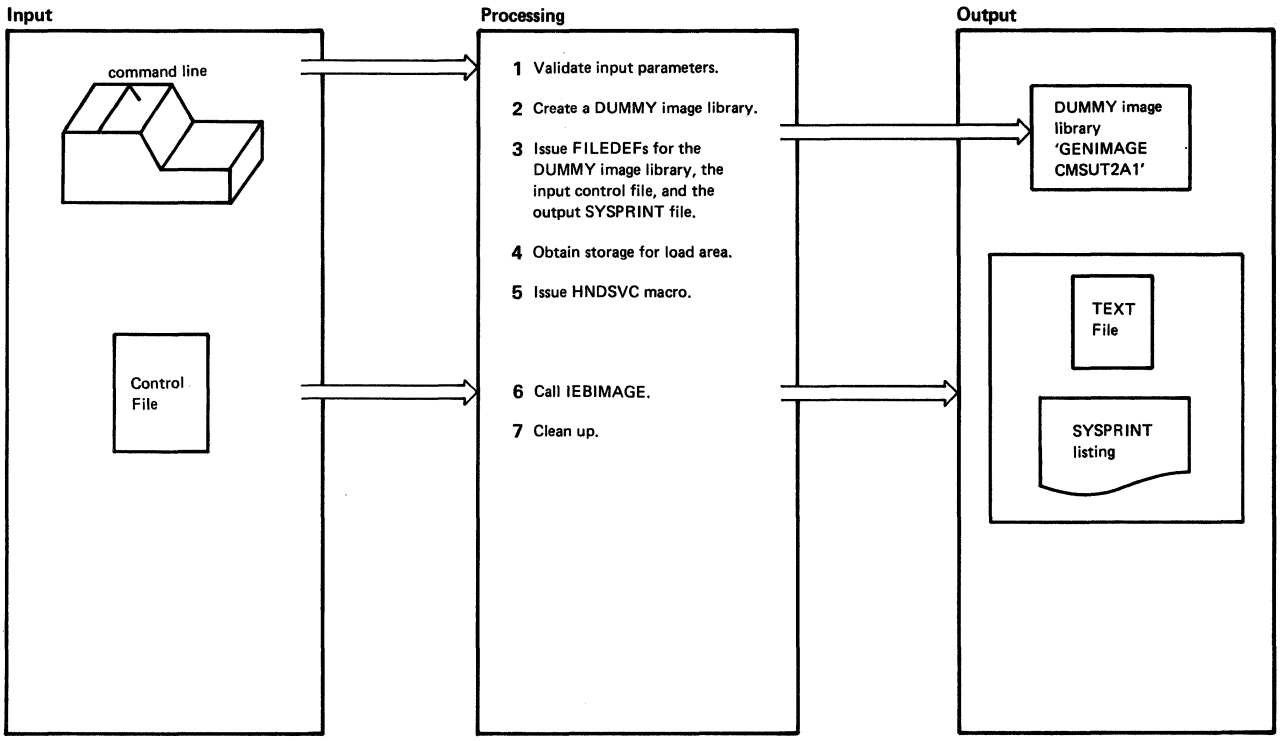
The IMAGELIB program (module DMKNMT) invoked by the IMAGELIB command, loads the necessary TEXT decks into the named system allocated at system generation time.

An installation can modify an existing 3800 named system using the IMAGEMOD command. This command is described in *VM/SP System Logic and Problem Determination Guide Volume 2 (CMS).*

Note: Due to the change in pel density, customized 3800 Model 1 character sets are not interchangeable with the 3800 Model 3 character sets. Users may recode customized 3800 Model 1 character sets and build new modules through the use of the GENIMAGE command. The MVS Character Conversion Aid may also be used to convert existing customized character sets to the 3800 Model 3 pel density.

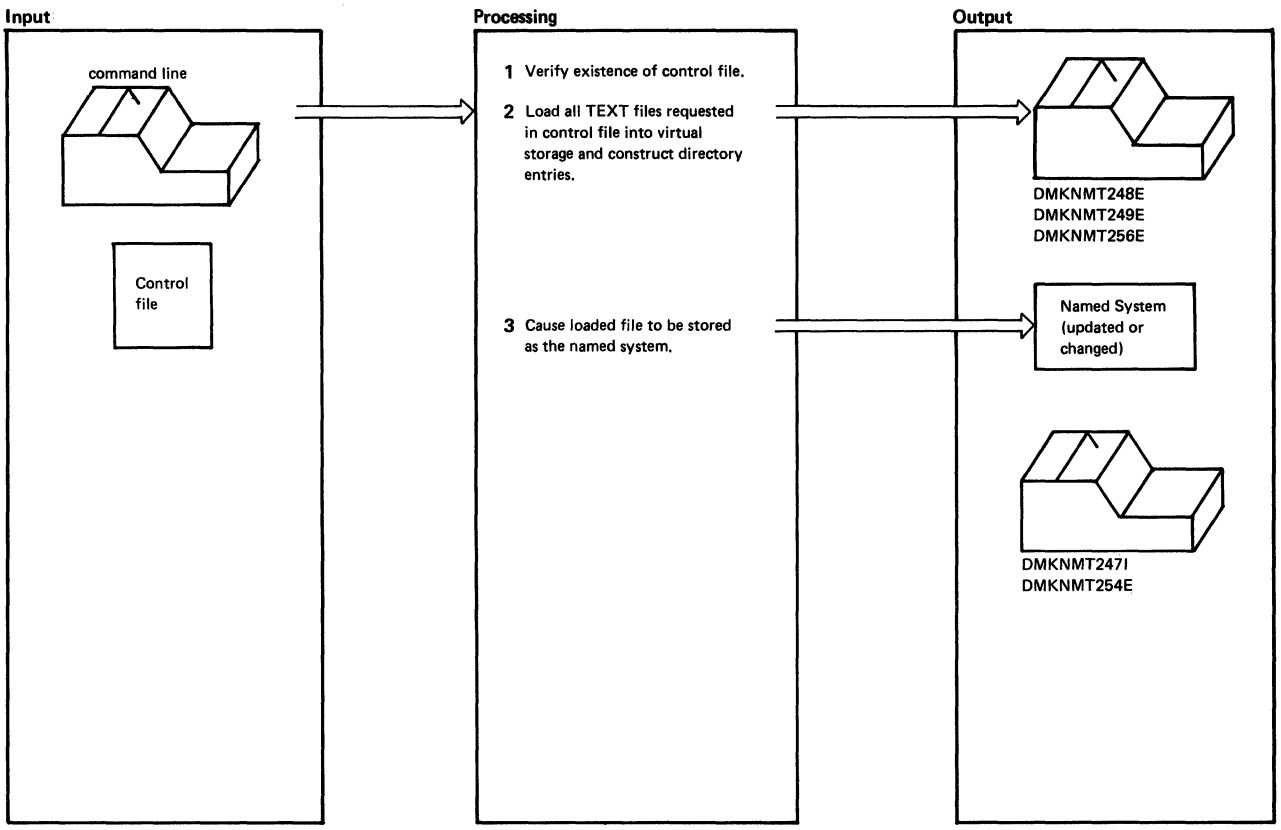
Method of Operation

This section describes modules DMKIMG and DMKNMT. Figures 93 and 94 describe the functions of these modules and serve as a guide to the program listings. The labels shown indicate the closest label to the function being documented. Use the Directory and Program Organization sections to find the labels in the program listings for any routines that are not shown in the Method of Operation section.



Notes		Module	Label	Ref
1	GENIMAGE command parameters are validated. If a parameter is invalid, issue return code 100.	DMKIMG	LOOP1 LOOP2 PARMERR	
2	Create a DUMMY image library.		ENDPARMS	
3	Issue a FILEDEF command with the AUXPROC option for the DUMMY image library created in Step 2; this traps all READ and WRITE operations on that data set. If any FILEDEF errors occur, issue a return code of 104.		FILEBAD	
4	Issue a GETMAIN for a 73,000 byte area for simulating OS LOAD macros.			
5	Issue HNDSVC macro to handle the following SVCs: ● SVC 8 (LOAD) ● SVC 18 (BLDL) ● SVC 21 (STOW)			
6	Call IEBIMAGE			
	● Issue a CMS STATE command for the TEXT file being searched for and set appropriate return codes in SVC save area.		BLDLRTN	
	● Issue CMS LOAD for requested module and return the address of the area loaded into, to the issuer of the LOAD command.		LOADRTN	
	● Use CMS LOAD command to get module into LOAD area and move data into user-supplied buffer for the READ.		READRTN	
	● Treat as no-op and return to issuer.		WRITERTN	
	● Simulate operation of STOW macro by locating the module data in the IEBIBLKS work area and create a TEXT deck from it: 1) Create ESD (external symbol directory) card and write to TEXT file (GENIMAGE CMSUT1) 2) Create all necessary TXT cards and write to TEXT file (GENIMAGE CMSUT1) 3) Create an END card and write to TEXT file (GENIMAGE CMSUT1)		READEXIT STOWRTN TXTLOOP	
7	Erase old TEXT file (if one existed) and rename GENIMAGE CMSUT1 to a TEXT file named IEBIMAGE.			

Figure 93. DMKIMG



Notes		Module	Label	Ref
1	Verify the existence of the control file. If it doesn't exist, give a return code of 4.	DMKNMT	IMAGELIB ERR004	
2	Create a DUMMY directory that will be used to hold the number of entries in the named system.			
	Read a record from the control file and verify the existence of the indicated TEXT file. If it doesn't exist, issue message DMKNMT248E.		RDLOOP AFTERRD NOTEXT	
	Load the TEXT file into the CMS transient area. If a LOAD error occurs, issue message DMKNMT256E.		LDERR	
	Move the file from the transient area to the core image area if sufficient storage exists. If not, issue message DMKNMT256E.			
	Create a new directory entry for this TEXT file and return to RDLOOP. If no more entries, close the control file, compress the core image, and adjust the displacements in the directory.		RANOUT RDEOF DSPLOOP	
3	Issue DIAGNOSE X'74' to cause the named system to be saved. If successful, issue message DMKNMT247I; if not successful, issue message DMKNMT254E.		DIAGERR	

Figure 94. DMKNMT

Program Organization

This section includes program descriptions of modules DMKIMG and DMKNMT.

DMKIMG

Provides a CMS interface for the VS-based IEBIMAGE program by handling certain SVCs issued by IEBIMAGE and translating them into CMS terms.

Entry Point

DMKIMGBG

Routines Called

Routine	Purpose
FSSTATE	Determines if control file exists
HNDSVC	Traps certain SVCs issued by IEBIMAGE
GETMAIN	Gets area for simulating OS LOAD SVC
FREEMAIN	Releases OS LOAD area
FILEDEF	Issues FILEDEFs needed by IEBIMAGE
LOAD	Simulates OS LOAD and QSAM READ
FSWRITE	Creates a new TEXT file (STPW simulation)

Attributes

Disk resident, loaded into CMS user area, called via SVC 202, serially reusable

Registers at Entry

Reg	Use
R1	Standard CMS PLIST
R14	Return address
R15	Address of GENIMAGE

Registers at Exit

Reg	Use
R15	Return code < 100 for normal IEBIMAGE execution
R15	Return code 100 if error in input parameters
R15	Return code 104 if error during FILEDEF

External References

MAINHIGH - Saves and restores its value between loads

DMKNMT

Constructs an image library from TEXT files on user disks and creates or replaces that image library via DIAGNOSE code X'74'.

Entry Point

DMKNMTBL

Routines Called

Routine	Purpose
FSSTATE	Determines if CNTRL and TEXT files exist
FSREAD	Reads in the control file
CMS LOAD	Loads the TEXT file into the transient area

Attributes

Disk resident as "IMAGELIB," loaded into CMS user area, called via SVC 202, serially reusable

Registers at Entry

Reg	Use
R1	Standard CMS PLIST

Registers at Exit

Register 15 contains a return code:

Return Code	Meaning
0	Image library updated successfully
4	Control file not found or in error
8	Specified image non-existent
12	Specified image caused LOAD error
16	Insufficient virtual storage
20	Image library is currently active
100	Error in FSREAD return code

Register Usage

Reg	Use
R0	Temporary work register
R1	PLIST register and temporary work register
R2	Source address for MVCL
R3	Source length for MVCL
R4	Target address for MVCL
R5	Target length for MVCL
R6	Current end of image library in storage
R7	Pointer to next available directory entry
R8	Running counter for number of directory entries
R9	Starting address of the image library in storage
R12	DMKNMT module base
R14	BALR return address and scratch register
R15	BALR branch address and scratch register

External References

None

Directory

The major labels in modules DMKIMG and DMKNMT are listed below in alphabetical order.

DMKIMG

Label	Figure	Description
BLDL2	93	Checks for file
BLDL3	93	
BLDRET	93	Return to user key
-ENDPARMS	93	Creates DUMMY image library
-FILEBAD	93	Issues FILEDEF error
GETSEQ	93	Obtains current value of sequence number
LOADRTN	93	Simulates LOAD functions
LOOP1	93	Validates parameter list
LOOP2	93	Validates options
MOVETXT	93	
OPTIONS	93	Scans through options
PARMERR	93	Gives return code 100 for parameter error
READEXIT	93	Issue return codes from READ
READRTN	93	Simulates READ functions
RETURN	93	Saves return code
STOWRTN	93	Simulates STOW functions
TXTLOOP	93	Creates TXT cards
WRITERTN	93	Simulates WRITE functions

DMKNMT

Label	Figure	Description
AFTERRD	94	Saves the name of the control file
DIAGERR	94	Issue error message DMKNMT254E
DSPLOOP	94	Adjusts old displacement in directory entries
ERR004	94	Issues return code of 4
LDERR	94	Issues error message DMKNMT249E
NOTEXT	94	Issues error message DMKNMT248E
RANOUT	94	Issue error message DMKNMT256E
RDEOF	94	Saves file name for CLOSE
RDERR	94	Checks for end-of-file
RDLOOP	94	Points to file name
RETURN	94	Obtains return address

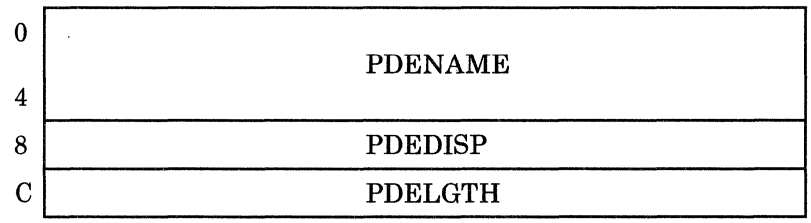
Data Areas

The following data areas are used by DMKIMG:

- Data Control Block (DCB)
- Data Extent Block (DEB)
- Data Extent Control Block (DECB).

The above data areas are described in the *OS/VS2 System Programming Library: Debugging Handbook Volume 2.*

DMKNMT uses PDEBLOK. The layout of a PDEBLOK Directory entry is described in Figure 95.



Displacement Hex	Dec	Field Name			Description
0	0	PDENAME	DS	CL8	Member name
8	8	PDEDISP	DS	1F	RBA of start of member
0C	12	PDELGTH	DS	1F	Length of member in bytes

Figure 95. PDEBLOK Directory Entry for Named System

Diagnostic Aids

Following is a list of the messages issued by DMKNMT. The nearest label and the associated method of operation diagram are identified.

Message Code	Label	Figure	Message Text
DMKNMT247I	RETURN	94	3800 NAMED SYSTEM imagelib 3800 CREATED
DMKNMT248E	NOTEXT	94	SPECIFIED IMAGE imagelib NON-EXISTENT
DMKNMT249E	LDERR	94	ERROR LOADING IMAGE imagelib
DMKNMT254E	DIAGERR	94	ERROR SAVING imaglib; RC = nn
DMKNMT256E	RANOUT	94	INSUFFICIENT VIRTUAL STORAGE
DMKNMT257E			RESIDUAL BYTE COUNT = nnnnnnnn (HEX)

Index

Chapter 12. Command Class Override

Command Class Override

Introduction

The DMKOVR program builds an internal class-override file on a volume previously formatted by the Format/Allocate program as type OVRD.

VM/SP is distributed with the CP commands and DIAGNOSE codes assigned to one or more of the eight privilege classes (see the *VM/SP Operator's Guide* or the *VM/SP CP Command Reference* for a list of what classes each command is assigned to.) Installations can redefine the assignment of privilege classes using up to 32 classes (A through Z and 1 through 6) to tailor the authorization structure of their system.

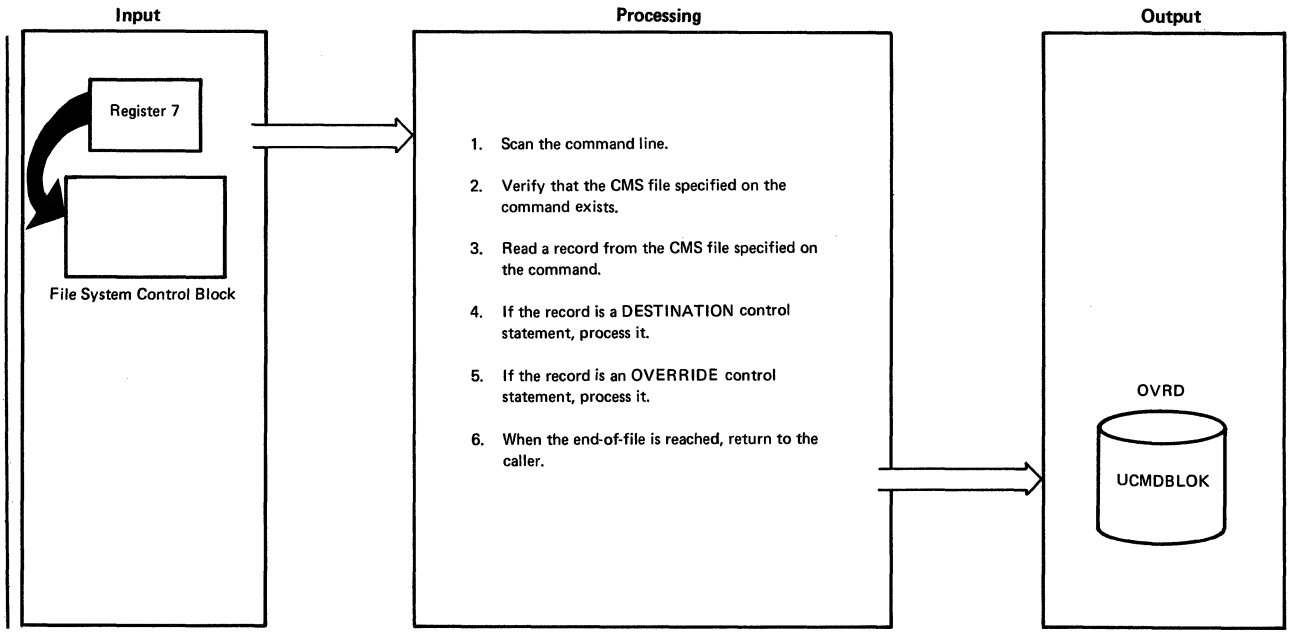
To redefine the privilege classes for certain commands, the user creates a CMS file that contains a DESTINATION control statement and an OVERRIDE control statement for each command whose class is being changed. (See the *VM/SP CP for System Programming* for a detailed description of these control statements and the steps to take.) The changes to privilege classes described by these control statements are activated when the user enters an OVERRIDE command.

When the user issues the OVERRIDE command, DMKOVR receives control. DMKOVR scans the parameters specified on the OVERRIDE command, one of which is the filename of the CMS file that contains the DESTINATION and OVERRIDE control statements. Using this file, DMKOVR builds a class-override file in internal format that describes the new privilege classes for the specified commands.

Method of Operation

This section describes those functions that the DMKOVR program performs. There is only one method of operation illustration and that is Figure 96 on page 264.

Command Class Override



Notes		Module	Label	Ref
1	Scan the OVERRIDE command.	DMKOVR	CKINP	
	A. The user must specify the filename and filetype for the CMS override file. If not, issue the message DMKOVR763E INVALID FILENAME OR FILE NOT FOUND Because there is no file to process, go to Step 6 to clean up and return to the caller.		ERR763	
	B. If the specified the FREE option, set an indicator (FREE) in OVRFLAG1. If the user specified the EDIT option, set an indicator (EDITMODE) in OVRFLAG1. If the user specified anything other than EDIT or FREE, issue the message DMKOVR751E INVALID OPERAND - operand Continue processing in edit mode. That is, scan the control statements for valid syntax but do not build an internal override file.		ERR751	
2	Issue that FSSTATE macro to determine if the CMS file specified on the OVERRIDE command exists. If the file does not exist, issue the message DMKOVR763E INVALID FILENAME OR FILE NOT FOUND Because there is no file to process, go to step 6 to clean up and return to the caller.	DMKOVR	ERR763	
3	Issue the FSREAD macro to read a record from the CMS file specified on the OVERRIDE command.	DMKOVR	USERREAD	
4	If the record is a DESTINATION control statement, process it as follows:	DMKOVR	DIRSTMT	
	A. If the DESTINATION control statement was immediately preceded by another DESTINATION control statement, issue the message DMKOVR751E INVALID OPERAND - operand Continue processing in edit mode. That is, scan the control statements for valid syntax but do not build an internal override file.			
	B. Use the first three parameters in the control statement (cuu devtype volser) to help locate the override space on the CP-owned volume. The user has allocated this space with the Format/Allocate program as type OVRD.			
	C. If the user specified the FREE option on the OVERRIDE command, set cylinder 0 record 3 on the CP-owned volume (offset 56) to blanks. This indicates that an override file does not exist.			
	D. If the user did not specify the FREE option, put the address of the override space in cylinder 0 record 3 on the CP-owned volume. Go to step 3 to read the next record from the CMS file.			

Figure 96 (Part 1 of 2). DMKOVE - Class Override Program Processing

Notes			Module	Label	Ref
5	If the record is an OVERRIDE control statement, process it as follows.		DMKOVR	OVRSTMT	
	A.	If a DESTINATION control statement has not been read, issue the message DMKOVR762E DESTINATION STATEMENT MISSING Continue processing in edit mode. That is, scan the control statements for valid syntax but do not build an internal override file.		ERR762	
	B.	Scan the control statement for the command name or DIAGNOSE code. Assume that the first nonblank character string in the command name or DIAGNOSE code. Compare this string to a table of valid commands and DIAGNOSE codes to ensure that it is valid. If it is not, issue the message DMKOVR751E INVALID OPERAND - operand Continue processing in edit mode. That is, scan the control statements for valid syntax but do not build an internal override file.		SCANCOM ERR751	
	C.	Scan the control statement for the TYPE keyword. Ensure that TYPE is not specified for DIAGNOSE code, that there is only one character following the TYPE keyword, and that the type is valid for the specified command. If any of these errors are found, issue the message DMKOVR751E INVALID OPERAND - operand Continue processing in edit mode. That is, scan the control statements for valid syntax but do not build an internal override file. If the type is valid, save it in TYPESAVE.		SCANKEY DOTYPE ERR751	
	D.	Scan the control statement for the CLASS keyword. Verify that the classes specified are valid command classes (A through Z or 1 through 6) and that no class is repeated. If either of these errors is found, issue the message DMKOVR765E INVALID CLASS DEFINITION - x or DMKOVR766E DUPLICATE CLASS DEFINITION - x Continue processing in edit mode. That is, scan the control statements for valid syntax but do not build an internal override file. Set the class mask in CLASBITS to indicate the specified classes. If the class is an asterisk (*), all classes are allowed for this command; set the class mask in CLASBITS to indicate all classes.		DOSCLASS ERR765 ERR766	
	E.	If the OVERRIDE control statement did not include the CLASS keyword, or if it included the TYPE keyword and there is more than one type for this command, issue the message DMKOVR753E OPERAND MISSING Continue processing in edit mode. That is, scan the control statement for valid syntax but do not build an internal override file.		ENDCARD ERR753	
	F.	Build the record to be placed in the internal override file. (If the user specified the EDIT option or if an error forced continuation in edit mode; skip this step; return to step 3 to read the next record.) Write the record in the UCMDBLOK. Go to step 3 to read the next record from the CMS file.		OVRBUILD	
6	If there are no more records in the CMS file, perform clean-up processing. If the user specified the FREE option on the OVERRIDE command or no options, write the volume label and allocation map, and issue the message EOJOVERRIDE FILE UPDATED If the user specified the EDIT option on the OVERRIDE command or if an error forced continuation in edit mode, issue the message EOJOVERRIDE FILE NOT UPDATED.		DMKOVR	WRTVOL FINISH	

Figure 96 (Part 2 of 2). DMKOVE - Class Override Program Processing

Program Organization

This section includes a program description of the DMKOVR module.

DMKOVR

Builds a command class-override file in internal format

Entry Points

DMKOVRDE

Routines Called

None

Attributes

Not serially reusable; nonresident.

Registers at Exit

Register 15 contains a return code at exit.

Return Code	Meaning
0	Override file successfully updated
4	Invalid operand or operand missing
8	I/O error loading the override file
12	Invalid option
20	Invalid character in file-id
24	Invalid filemode
28	Invalid filename or file not found
36	Disk not accessed

where xx is the CMS routine return code

Return Code	Meaning
1xx	Error in the CMS RDBUF routine
2xx	Error in the CMS TYPLIN routine

Command Class Override

Register Usage

Reg	Use
R0	Not used
R1	Work register
R2	Work register from scanning routine
R3	Work register
R4	Work register
R5	Work register
R6	Work register
R7	Work register
R8	Work register
R9	Pointer to device table entry
R10	Base address for UHDRBLOK
R11	Base register for work areas and constants
R12	Base register for code
R13	Base address for save area
R14	Return address to CMS; linkage to subroutines
R15	Entry address; on exit contains a return code

External References

None

Directory

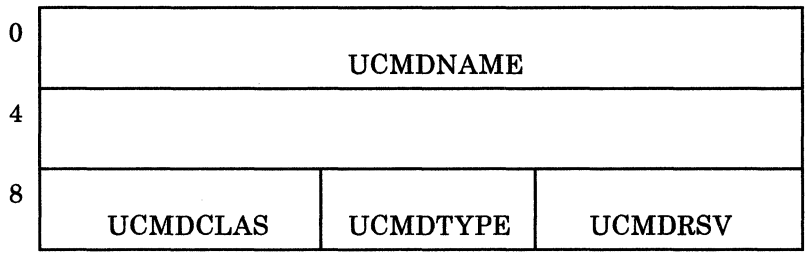
Following is an alphabetic list of the major labels of the class override program. The list references the associated method of operation diagram and includes a brief description of the function performed at the point in the program corresponding to each label.

Label	Figure	Description
CHEKEOF		Receives control when there is an error reading a record from the CMS file
CKINP	96	Scans the OVERRIDE command line
DIRSTMT	96	Processes the DESTINATION control statement
DMKOVRDE		Sets up base registers and initializes pointers
DOCLASS	96	Processes the CLASS keyword on the OVERRIDE control statement
DOIO		Performs DASD I/O such as writing a directory page and writing the volume label and allocation map
DOTYPE	96	Processes the TYPE keyword on the OVERRIDE control statement
ENDCARD	96	Verifies that all required fields were specified on the OVERRIDE control statement
ERR751	96	Issues message DMKOVR751E
ERR753	96	Issues message DMKOVR753E
ERR754		Issues message DMKOVR754E
ERR755		Issues message DMKOVR755E
ERR760		Issues message DMKOVR760E
ERR761		Issues message DMKOVR761E
ERR762	96	Issues message DMKOVR762E
ERR763	96	Issues message DMKOVR763E
ERR764		Issues message DMKOVR764E
ERR765	96	Issues message DMKOVR765E
ERR766	96	Issues message DMKOVR766E
EXIT		Returns to the caller
FINISH	96	Determines whether to issue a message stating that the override file was updated or not updated
GETPAGE		Gets the next page to use for the internal override file
OVRBUILD	96	Builds the entry for the internal override file
OVRSCAN		Scans the OVERRIDE control statement for the TYPE and CLASS keywords
OVRSTMT	96	Processes the OVERRIDE control statement
OVRVER1		Verifies that the command or DIAGNOSE code exists
SCANCARD		Scans a record read from the CMS file, stopping at the first blank it encounters and converting alphabetic characters to uppercase
SCANCOM	96	Scans the command or DIAGNOSE code specified on the OVERRIDE control statement, ensuring that it is valid
SCANCUU		Scans a four-byte field, verifying that the first three bytes are hexadecimal characters and that the fourth byte is blank
SCANDEV		Scans the table of devices (DEVTAB) to locate the address of the device specified on the DESTINATION control statement
SCANKEY	96	Scans the OVERRIDE control statement for the TYPE and CLASS keywords and verifies that the keyword is followed by an equal sign

Label	Figure	Description
UPALLOC		Scans through the allocation table, releasing the old override cylinder (if any) and locating the next available cylinder. Updates the volume label record with the pointer to the cylinder that will contain the internal override file
USEREOF		Receives control when all records have been read from the CMS file
USERREAD	96	Reads a record from the CMS file
WRTVOL	96	Writes the volume label and allocation map

Data Areas

This section describes the UCMDBLOK DSECT that is used to map the internal class override records in the override space.



Displacement Hex	Dec	Field Name			Description
0	0	UCMDNAME	DC	CL8' '	Command or DIAGNOSE code name
8	8	UCMDCLAS	DC	XL4'00'	Class overrides
A	10	UCMDTYPE	DC	BL8'0'	Functional group type
B	11	UCMDRSV	DC	XL3'0'	Reserved

Figure 97. UCMDBLOK DSECT

Diagnostic Aids

Following are the messages issued by the class override program. The list includes the label of the message and the associated method of operation figure.

Message Code	Label	Figure	Description
DMKOVR751E	MSG751	96	INVALID OPERAND - operand
DMKOVR753E	MSG753	96	OPERAND MISSING
DMKOVR754E	MSG754		DEV rdev NOT OPERATIONAL
DMKOVR755E	MSG755		IO ERROR rdev CSW = csw SENSE = sense
DMKOVR760E	MSG760		NOT ENOUGH SPACE ALLOCATED FOR OVERRIDES
DMKOVR761E	MSG761		VOLID READ IS volid1 NOT volid2 ON rdev
DMKOVR762E	MSG762	96	DESTINATION STATEMENT MISSING
DMKOVR763E	MSG763	96	INVALID FILENAME OR FILE NOT FOUND
DMKOVR764E	MSG764		ERROR IN routine
DMKOVR765E	MSG765	96	INVALID CLASS DEFINITION - x
DMKOVR766E	MSG766	96	DUPLICATE CLASS DEFINITION - x
	MSGEOK		EOJ OVERRIDE FILE UPDATED
	MSGEBAD		EOJ OVERRIDE FILE NOT UPDATED

Command Class Override

Index

Command Class Override

Summary of Changes

To obtain editions of this publication that pertain to earlier releases of VM/SP, you must order using the pseudo-number assigned to the respective edition. For:

- Release 4, order LT00-1602
- Release 3, order LQ60-0890
- Release 2, order LT60-0890 (updated by TNL LN24-5714)
- Release 1, order LT60-0890

Summary of Changes for LY20-0890-3 for VM/SP Release 5

This revision includes:

- Reorganization of front matter information.
  - The summary of changes has been moved to page 275.
  - The list of terms used in this book has been moved to page 277.
  - The list of related publication has been moved to page 279.
- Changes to message texts.
- Page layout and format has been changed.
- Each service routine is made to appear more as a unique entity by providing a table of contents and an index for each service routine.

It also includes minor technical and editorial changes.

Summary of Changes for LY20-0890-2 for VM/SP Release 4

This revision includes changes for:

- DMKOVR - a new command class override program
- A new option, COMPACT, for the DDR function
- IBM 3800 Model 3 compatibility support
- IBM 3480 Magnetic Tape Subsystem support.

It also includes minor technical and editorial changes.

Summary of Changes for LY20-0890-1 for VM/SP Release 3

This revision includes changes for the EREP/Error Recording Interface. It also includes minor technical and editorial changes, and removes documentation for the Interactive Problem Control System and the LKED command processor.

Summary of Changes for LY20-0890-0 as updated by LN24-5714

This revision includes changes for the IBM 3375 and 3380 Direct Access Storage Devices and for the Directory Maintenance program product. It also includes minor technical and editorial changes. Documentation for the IBCDASDI program and the GENERATE procedure is removed.

Glossary of Terms and Abbreviations

In this publication:

The term "2305 series" is used in reference to the IBM 2305 Disk Storage, Models 1 and 2.

The term "3330 series" is used in reference to the IBM 3330 Disk Storage, Models 1, 2, and 11, and the IBM 3333 Disk Storage and Control, Models 1 and 11; and the 3350 Direct Access Storage operating in 3330/3333 Model 1 or 3330/3333 Model 11 compatibility mode.

The term "3340 series" is used in reference to the IBM 3340 Disk Storage, Models A2, B1 and B2, and the 3344 Direct Access Storage Model B2.

The term "3350 series" is used in reference to the IBM 3350 Direct Access Storage Models A2 and B2 in native mode.

The term "3370" refers to the IBM 3370 Direct Access Storage Models A1, A2, B1, and B2.

The term "3375" refers to the IBM 3375 Direct Access Storage Facility.

The term "3380" refers to the IBM 3380 Storage Facility.

The term "3480" refers to the IBM 3480 Magnetic Tape Subsystem.

The terms "3705," "370X," and "3704/3705" include the IBM 3704, 3705-I, and 3705-II Communications Controllers, unless otherwise specified.

The term "3800" refers to the IBM 3800 Printing Subsystems, Models 1, 3, and 8. A specific device type is used only when a distinction is required between device types. References to the 3800 Model 3 apply to both Models 3 and 8 unless otherwise explicitly stated. The IBM 3800 Model 8 is available only in selected world trade countries.

The term "3880" refers to the IBM 3880 Storage Controller.

The term "FB-512" (FBA) is used in reference to the IBM 3310 and 3370 Direct Access Devices.

Any mention of the IBM 2741 Communication Terminal also applies to the IBM 3767 Communication Terminal, unless otherwise stated.

Bibliography

Refer to the following publications for related material:

VM/SP Operator's Guide, SC19-6202

VM Diagnosis Guide, LY24-5241

VM/SP CP for System Programming, SC24-5285

VM/SP CMS Command Reference, SC19-6209

VM/SP CP Command Reference, SC19-6211

VM/SP Data Areas and Control Block Logic Volume 1 (CP), LY24-5220

VM/SP Data Areas and Control Block Logic Volume 2 (CMS), LY24-5221

VM/SP Planning Guide and Reference, SC19-6201

VM/SP Installation Guide, SC24-5237

VM/SP System Logic and Problem Determination Guide Volume 1 (CP), LY20-0892

VM/SP System Logic and Problem Determination Guide Volume 2 (CMS), LY20-0893

VM/SP System Messages and Codes, SC19-6204

VM/SP System Messages Cross-Reference, SC24-5264 *EREP User's Guide and Reference*, GC28-1378

OS/VS1 Programmer's Reference Digest, GC24-5091

OS/VS1 System Data Areas, SY28-0605

OS/VS2 System Programming Library: Debugging Handbook Volume 1, GC28-1047

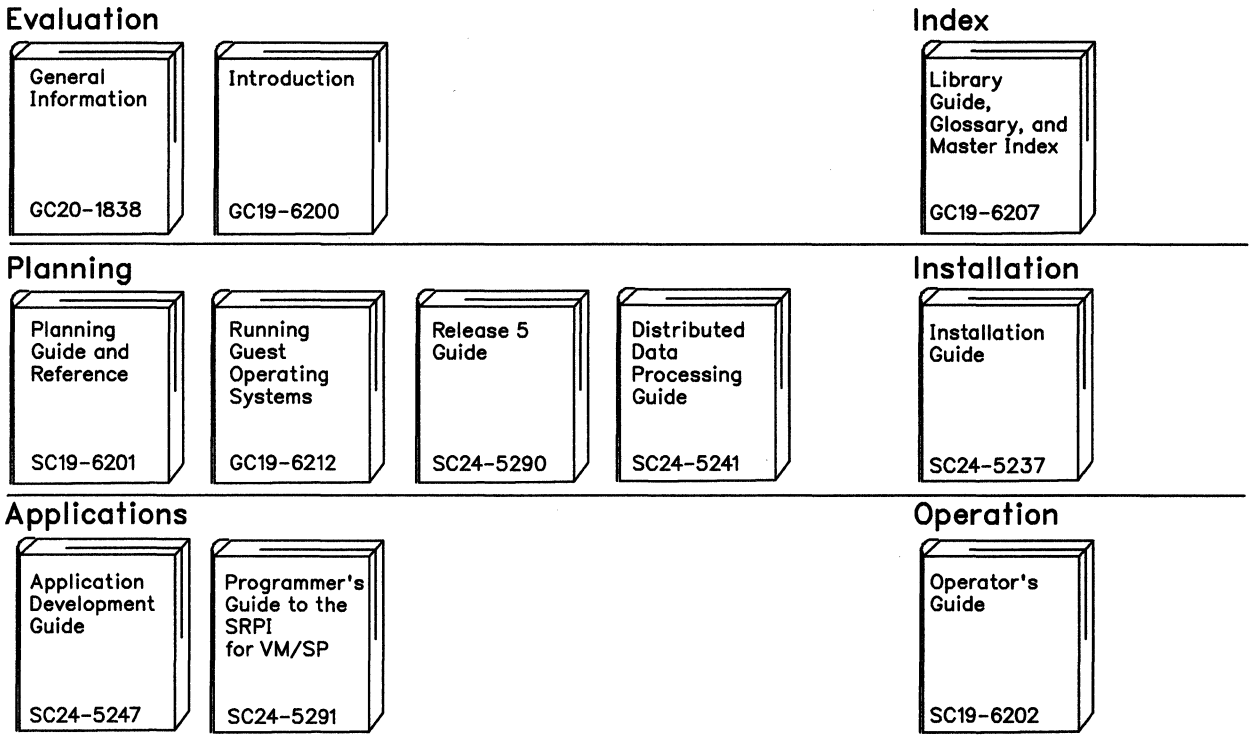
OS/VS2 System Programming Library: Debugging Handbook Volume 2, GC28-1048

OS/VS2 System Programming Library: Debugging Handbook Volume 3, GC28-1049

Concepts of the IBM 3800 Printing Subsystem, GC20-1775

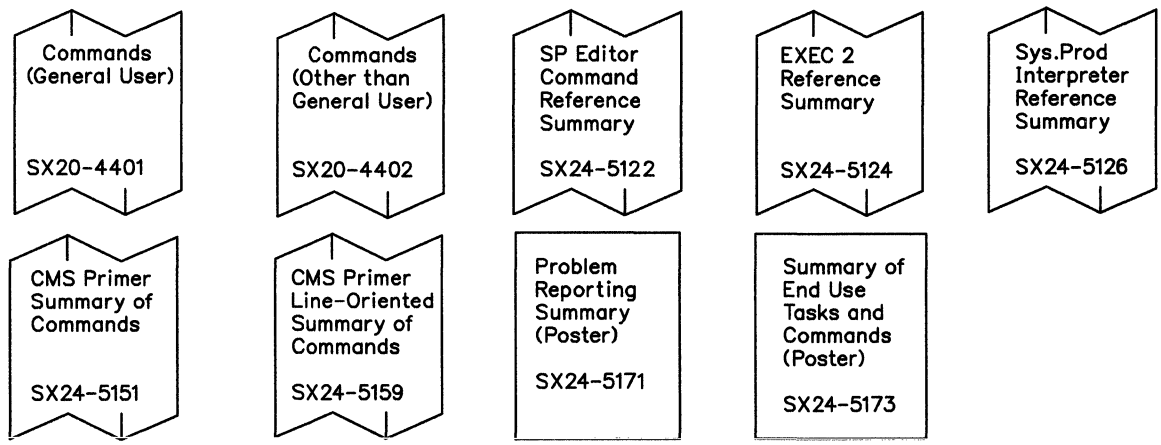
Reference Manual for the IBM 3800 Printing Subsystem, GA26-1635

The VM/SP Library (Part 1 of 3)



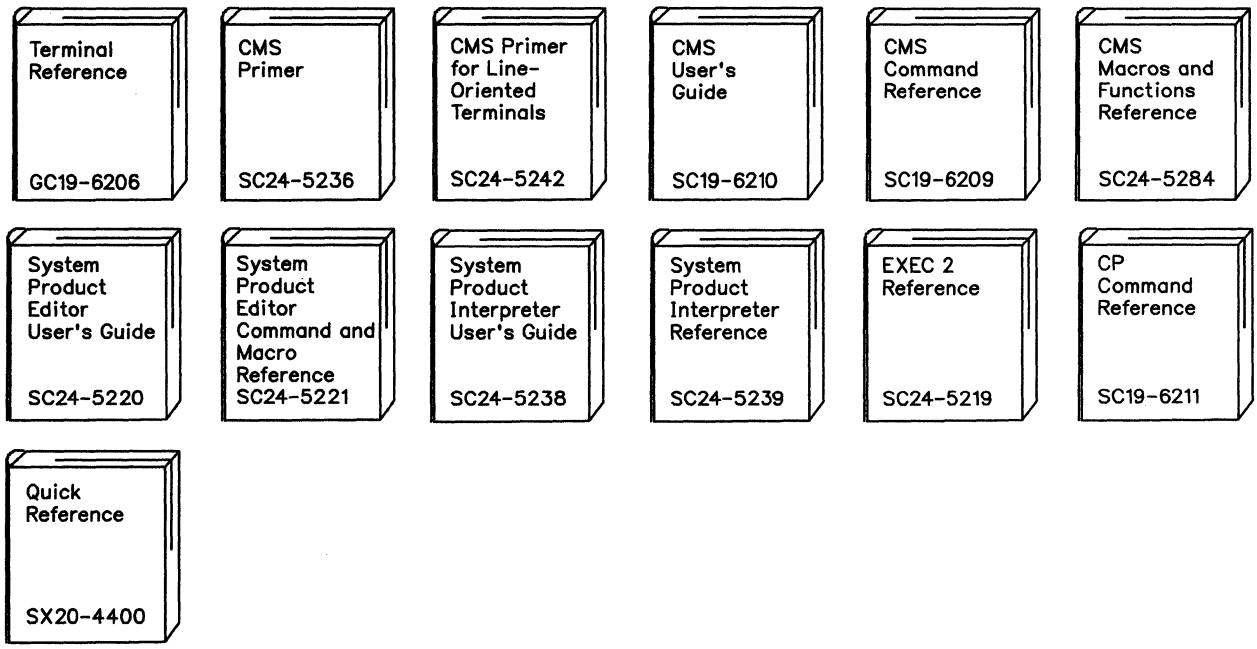
Reference Summaries

To order all of the Reference Summaries, use order number SBOF-3242

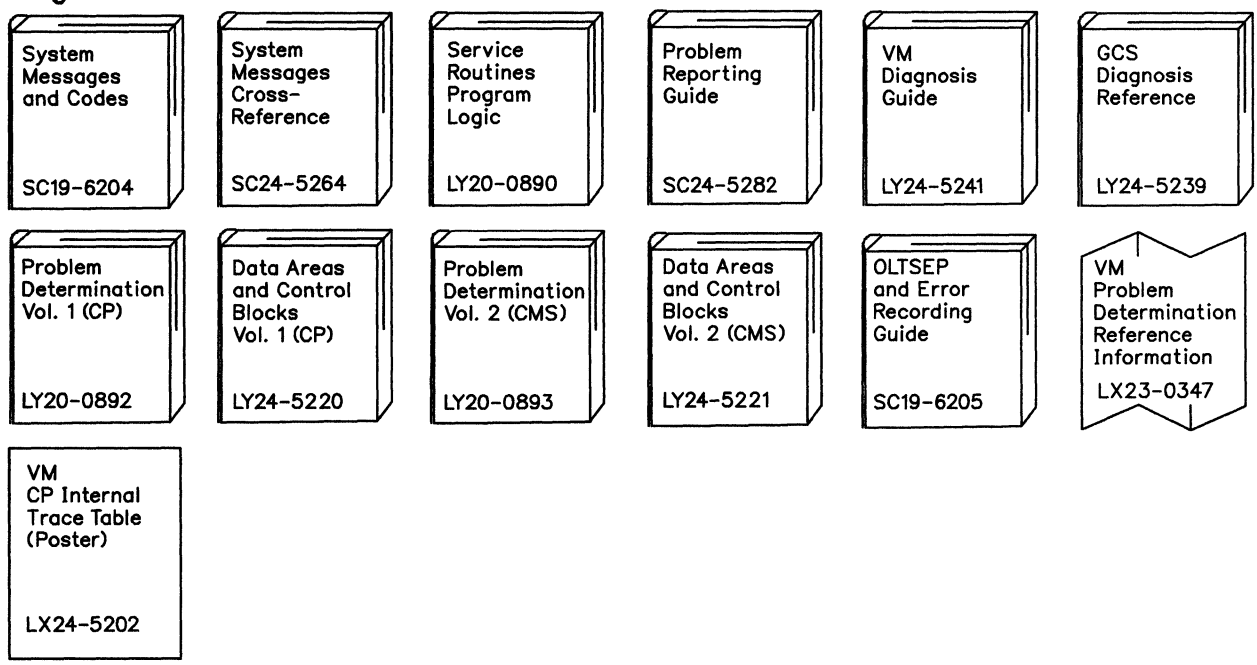


The VM/SP Library (Part 2 of 3)

End Use

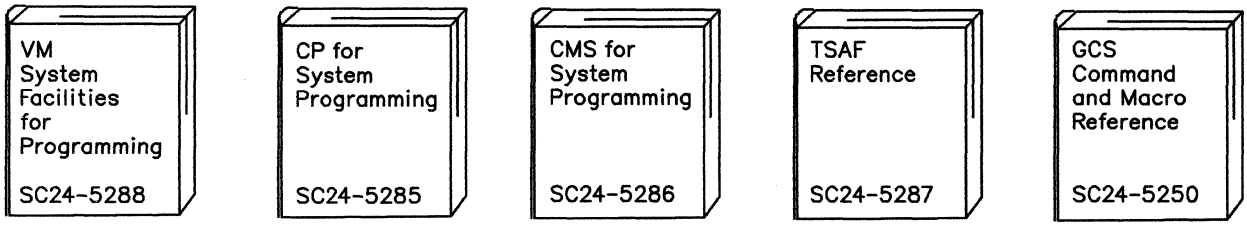


Diagnosis

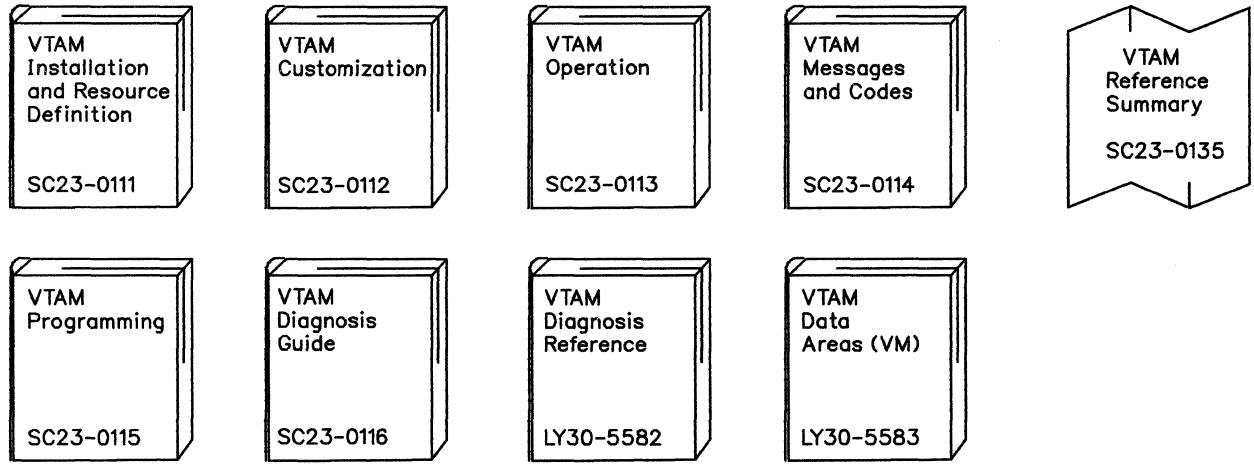


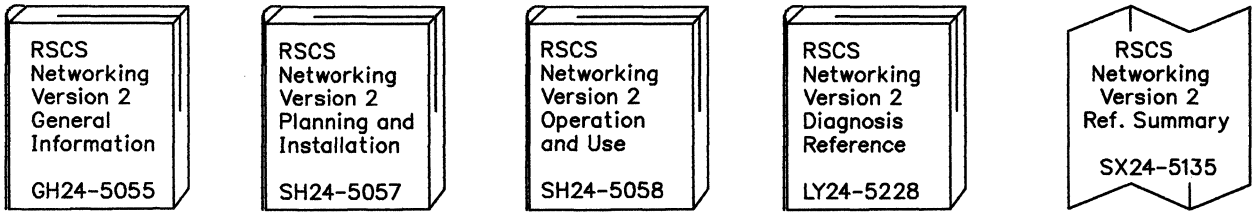
The VM/SP Library (Part 3 of 3)

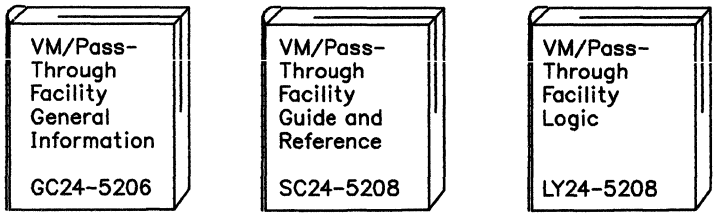
Administration



Auxiliary Communication Support







Index

Each chapter in this book is a complete unit, unrelated to other chapters except for all of them being utilities. Therefore, each chapter has its own index.

International Business
Machines Corporation
P.O. Box 6
Endicott, New York 13760

File No. S370/4300-37
Printed in U.S.A.

LY20-0890-3

VM/SP Service Routines
Program Logic
Order No. LY20-0890-3

READER'S COMMENT FORM

Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

If you use this form to comment on the online HELP facility, please copy the top line of the HELP screen.

_____ _____ _____ **Help Information line ____ of ____**

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered nonconfidential.

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply? ___YES ___NO

Please print your name, company name, and address:

_____

_____

_____

_____

IBM Branch Office serving you: _____

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.

LY20-0890-3
File No. S370/4300-37

Reader's Comment Form

CUT
OR
FOLD
ALONG
LINE

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT G60
PO BOX 6
ENDICOTT NY 13760-9987

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

International Business Machines Corporation
P.O. Box 6
Endicott, New York 13760

File No. S370/4300-37
Printed in U.S.A.

LY20-0890-3